Қазақстан Республикасының Білім және ғылым министірлігі
А. Байтұрсынов атындағы Қостанай мемлекеттік университеті
Бағдарламалық қамтамасыз ету кафедрасы

Ж. З. Сатмаганбетова

БАҒДАРЛАМАЛАУ ТЕХНОЛОГИЯЛАРЫ

Оқу құралы

Қостанай, 2017

УДК 004.42(075.8)
ББК 32.973-018я73
С 23
Құрастырушы: Сатмаганбетова Жанар Зарлыканқызы, бағдарламалық қамтамасыз ету кафедрасының аға оқытушысы

Пікір жазғандар:
Жунусов К.М. М. Дулатов атындағы ҚИнЭУ ақпараттық технологиялар мен автоматика кафедрасының меңгерушісі, э.ғ.к., доцент
Исмаилов А.О., А. Байтұрсынов атындағы ҚМУ оқу үрдісін жоспарлау және ұйымдастыру басқармасының директоры, бағдарламалық қамтамасыз ету кафедрасының доценті, т.ғ.к.
Салыкова О.С., А. Байтұрсынов атындағы ҚМУ бағдарламалық қамтамасыз ету кафедрасының меңгерушісі, т.ғ.к., доцент

С 23. Сатмаганбетова Ж. З. Бағдарламалау технологиялары. Оқу құралы. – Қостанай: А. Байтұрсынов атындағы ҚМУ, 2017.- 132 бет

Бағдарламалу технологиялары пәні бойынша оқу құралында бағдарламалау жүйелерінің және технологияларының теориялық негіздері және лабораторлық жұмыстарды орындауға арналған әдістемелік нұсқаулар берілген. берілген. Бұл жұмыс: есептеуіш техникасы және бағдарламалық қамтамасыз ету, ақпараттық жүйелер және информатика мамандықтарының төменгі курс студенттеріне арналған.

УДК 004.42(075.8)
ББК 32.973-018я73

ISBN 978-601-7387-74-7

А. Байтұрсынов атындағы ҚМУ оқу - әдістемелік кеңесімен бекітілген,
26 .04 . 2017 ж. № 3 хаттама

Мазмұны

Кіріспе

Қазіргі заманда компьютерлік технологиялар ең тез дамып келе жатқан және жоғары кіріс әкелетін сала. Елу дамыған мемлекеттер арасына кіру мақсатын орындау үшін ең жаңа технологияларды ендіруге және дамытуға дайын ғалымдар өте қажет. Бүгінде ғылымның және өндірістің бір де бір саласын, мейлі ол ғарыштық технологиялар, мейлі мұнай немесе атом станциялары болса да, микропроцессорлық техникасыз елестетуге болмайды.

Заманауи қоғамның негізгі талабы – студенттерге ақпараттық технологиялар негіздерін үйрету, логикалық - құрылымдық ойлау қабілеттерін дамыту, ақпараттық технологияны өзіндік даму мен оны іске асыру құралы ретінде пайдалану дағдыларын қалыптастырып, ақпараттық қоғамға бейімдеу.

Ақпараттық технологиялар саласының болашақ мамандары компьютердің құрылысын, бағдарламалар жүйесін және оны пайдаланудың бар мүмкіндіктерін ғана меңгерумен шектелмей, бағдарламалау технологиясының негізін білу өте қажет.

Бағдарламаны жасау барысында көптеген қателер кетуі де мүмкін. Олар жөндеу кезінде анықталып түзетіледі, мұндай қателерді түзеу кейде ұзаққа созылады, сондықтан да бағдарламаны жазу өте күрделі және әрі қымбат жұмыс.

Бағдарлама құруды жеңілдету үшін әртүрлі әдіс–тәсілдер пайдаланылады. Оның бірі көлемді бағдарламаларды бірнеше бөлшектерге(модулдерге) бөледі де оларды рет – ретімен орындайды. Шағын жинақталған бағдарламалар модулдерін жеке тіркеп, стандартты модулдерге арналған арнайы бағдарламалар кітапханасында сақтауға болады.

Бағдарлама құруды жеңілдету үшін, алдымен оның алгоритмдік схемасын сызу керек, содан кейін бағдарламалау тілінде оның логикалық құрылымын жазады. Алгоритмдік схеманың жәрдемімен ақпаратты өндеудің егжей – тегжейлі үрдісі беріледі және орындалатын операциялардың реті анықталады. Алгоритмдік схеманы жасауға арнайы шартты белгілер қолданылады.

Басылымға ұсынылып отырған оқу құралында бағдарламалау тілдерінің және технологияларының теориялық негізі және C/C+ және Паскаль жоғары деңгейлі бағдарламалау тілдерінде жұмыс істеу ерекшеліктері берілген.

Бөлім - I. Бағдарламалау технологияларының негіздері

1 Алгоритмдеу негіздері

1.1 Алгоритмдік құрылымдар және құру әдістері

Егер сіз берілген есепті шешу үшін қандай да бір бағдарламалау тілінде бағдарлама жазғыңыз келсе, онда алдымен есепті шешудің алгоритмін құруыңыз керек. Алгоритм – информатикадағы ең іргелі ұғымдардың бірі.

Алгоритм деп берілген есепті шешу үшін жасалатын әрекеттерді дәл және қарапайым етіп жазуды айтамыз. Басқаша айтқанда, алға қойылған мақсатқа жетуде немесе берілген есепті шешуде орындаушыға біртіндеп қандай әрекеттер жасау керектігін дәл көрсететін нұсқауларды немесе іздеп отырған нәтижені алу мақсатында деректермен атқарылатын әрекеттердің орындалу реттілігін анықтайтын жарлықты алгоритм дейміз.

Алгоритм сөзі - IX ғасырда өмір сүрген, Орта Азияның ұлы ғалымы - *Әл-Хорезмидің* есімімен байланысты, латынша жазылуы *algorithmi* сөзінен алынған. Ол өзінің «Арифметикалық трактат» деген еңбегінде арифметикалық амалдарды орындау тәртібін ұсынған. Осылайша алгоритм ұғымы алдымен математикада ертеден қолданыла бастаған, кейін информатиканың теориялық объектісі ретінде XX ғасырдың 30-шы жылдарында зерттеле бастады.

Алгоритм ұғымын және оның қасиеттерін зерттейтін информатиканың бөлімін *алгоритмдер теориясы* деп атайды. Алгоритмдер теориясының жүйелі түрде зерттеле бастауын А.А.Черчтың 1936 жылы басылып шыққан жұмысымен (Черч тезисі) байланыстырады[1].

Сонымен, арифметикалық амалдарды орындау ережесі, геометриялық фигураларды салу ережесі, сөздердің жазылуының грамматикалық ережесі т.с.с. алгоритм деп аталып кеткен. Жалпы *«Алгоритм»* сөзі мағынасы: нұсқау, жарлық, рецепт, ереже, тәртіп, заң, жоба сөздеріне синоним болып келеді.

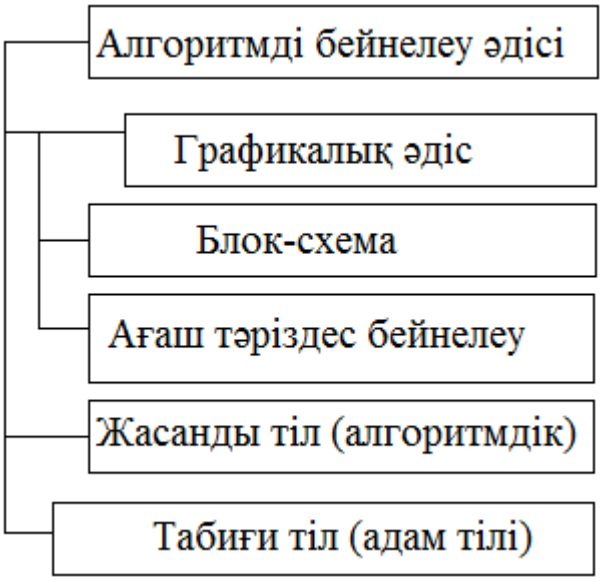
Алгоритм белгілі бір реттілікпен бірінен соң бірі орындалатын бірнеше қадамдардан тұрады. Алгоритмнің әрбір қадамы бір немесе бірнеше қарапайым операцияларды қамтиды. Алгоритм ұғымның мәнін аша түсетін оның мынадай қасиеттері бар:

1. Алгоритм айқын, дәл өрнектелген болу керек, яғни командасы орындаушыға түсінікті болуға тиіс. Бұл қасиеті анықтылық деп аталады.
2. Алгоритм модульдерге бөлінуі, яғни үлкен алгоритмді бірнеше ықшам жеке алгоритмдерге бөлген жөн. Бұл қасиеті дискреттілік деп аталады.
3. Алгоритм нақты бір мәселеге ғана құрылмай, осы тәріздес мәселелердің шешуін толық қамтуға бейімделіп құрылуы кажет. Бұл қасиеті жалпылық немесе ортақтық деп аталады.
4. Кез келген алгоритмнің міндетті түрде нәтижесі болуы керек, бұл қасиеті нәтижелік деп аталады.

Осы айтылғандардан алгоритм - бастапқы деректерді пайдаланып іздеген нәтижеге қол жеткізетін, реттелген әрекеттер тізбегі деген қорытынды жасауға

болады. Мұндай әрекеттер тізбегінің орындалуы алгоритмдік процесс, ал әрбір әрекет оның қадамы, әрбір нұсқау алгоритмнің қалыптасуы болып табылады.

Алгоритмнің құрамы дараланып және оның әрекеттері анықталғаннан кейін алгоритмді жазып көрсету тәсілін және тілін білу керек. Алгоритмді жазудың бірнеше тәсілдері бар. Төменде алгоритмді бейнелеу әдістерінің логикалық құрылымы көрсетілген(Сурет 1).



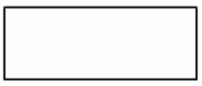
Сурет 1. Алгоритмнің берілу түрлері

Алгоритмді бейнелеу әдістерінің ішінен біз блок– схеманы пайдаланамыз. Блок схема компьютерге бағдарламалар жасау тәжірибесінде кеңінен қолданылатын алгоритмдерді жазудың графикалық тәсілі, басқаша айтқанда, алгоритмнің логикалық құрылымын график түрінде бейнелейтін тіл десек болады. Есепті шешу алгоритімінің блок-схемасын құрған кезде есепті шығару процесі кезеңдерге бөлініді. Әрбір кезең есептелетін операцияның сипатына байланысты белгілі конфигурациясы бар бір геометриялық фигурамен (блокпен) белгіленеді.

Блок деп аталатын мұндай фигуралардың ішіне кезеңдердің мазмұны жазылады. Есептелу процесінің бағыты блоктарды қосатын бағытталған кесіндімен көрсетіледі. Осы аталғандардың бәрі блок – схема тілінің алфавитін құрайды және олардың мағынасы алдын-ала келісілген келісім бойынша беріледі.

Атқаратын қызметі жағынан блоктар негізгі және қосымша болып бөлінеді. Негізгі блоктар енгізіу мен баспаға шығару және ақпараттарды өңдеу әрекеттерін білдіреді, ал қосымша блоктар схеманы түсіндіру және байланыстарды таңбалау үшін пайдаланылады. Алгоритмді блок-схема түрінде жазуда қолданылатын геометриялық фигуралар (Кесте 1) .

Кесте 1. Блок-схема фигуралары

№	Таңбның атауы	Таңба	Атқаратын қызметі
1	Басы-соңы (кіру-шығу)		Алгоритмнің басы мен соңын, көмекші программаға кіру мен шығу
2	Есептеу блогы (процесс)		Есептеулер немесе есептеулер тізбегі (ақпаратты өңдеу)
3	Логикалық блок		Шартқа байланысты алгоритмнің орындалу бағытын таңдау
4	Енгізу-шығару блогы		Берілген деректерді енгізу және нәтижені жауапқа шығару
5	Циклдік немесе параметрлі қайталану блогы		Қайталану процесстерінің саны белгілі болған жағдайда қолданылады
6	Нұсқама		Байланыс бағытын көрсету

Алгоритмнің базалық құрылымдары

Бұл элементар блок-сызбалардан алгоритмдеу тәжірибесінде қолданылатын үш базалық құрылым құрастырылады: *сызықтық, тармақталу, қайталану(циклдік).*

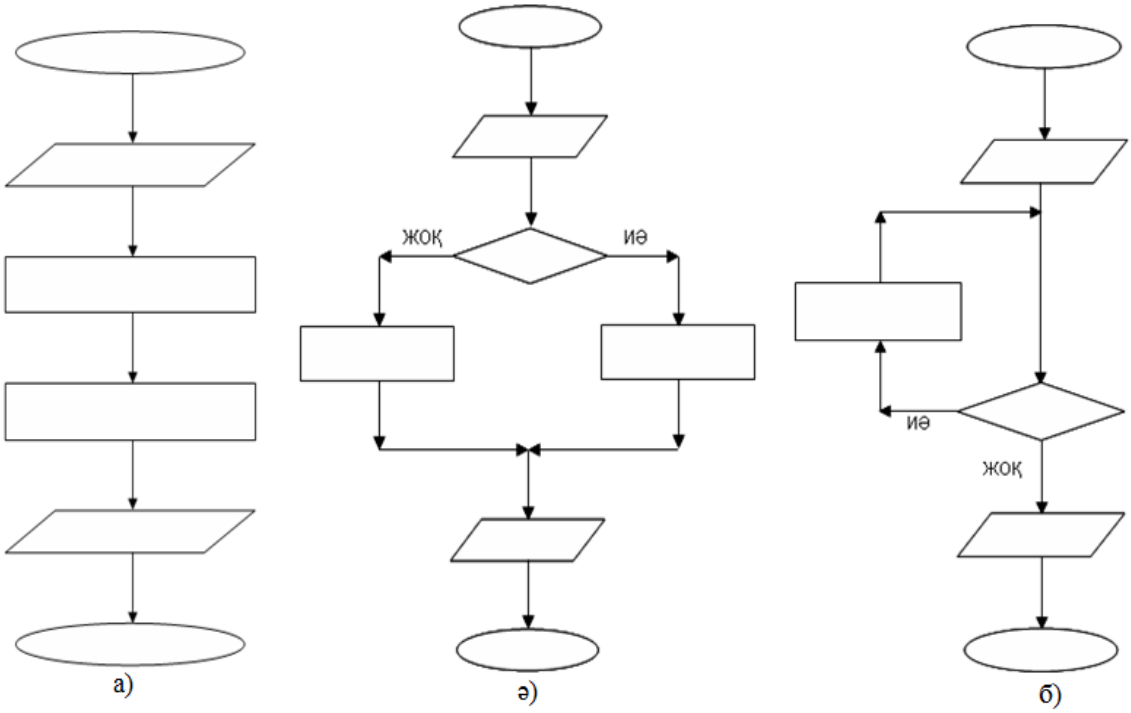
Сызықтық алгоритм. Сызықты алгоритм тізбектеле орналасқан командалардан, ал блок-схемасы бір сызық бойына орналасқан тізбекті блоктардан тұрады. Сызықтық алгоритмнің блок–схемасы мына суретте бейнеленген(Сурет 2а).

Тармақталу алгоритмдері. Егер алгоритм қадамдарының тізбектеле орындалуы қандай да бір шартқа тәуелді өзгеретін болса, онда ондай алгоритмді тармақталу алгоритм дейміз. Тармақталу алгоритмнің блок–схемасы мына суретте бейнеленген(Сурет 2ә). Тармақталу құрылымының өзі шарттың қойылуына байланысты толық, қысқа, таңдау болып үшке бөлінеді(Сурет 3/Қосымша А).

Қайталану(циклдік) құрылым. Алгоритмде белгілі бір әрекеттер тізбегі бірнеше рет қайталанатын болса, онда алгоритм циклдік деп аталады. Циклдік алгоритмнің блок–схемасы(Сурет 2б). Орындалатын командалар сериясын цикл денесі дейді.

Қайталану құрылымы команданың қойылуына байланысты әзірше, дейін, үшін деп үш түрге бөлінеді(Сурет 4/ Қосымша А). Қайталанбалы саны белгісіз,

берілген шартқа байланысты орындалатын циклдік құрылымдар: «Дейін», «Әзірше» деп аталады.



Сурет 2. Алгоритмнің базалық құрылымдары

Күрделі алгоритмдерді құру әдістері

Берілген есептің шартыны байланысты өңделетін ақпараттың күрделі болып келуі де мүмкін. Оны өңдеу алгоритмдік, жобалық тәсілдермен орындалуы мүмкін. Алгоритмді құрастыруды көптеген жолдары бар, бірақ ыңғайлы болатын, әрі ең түсінікті тәсілді таңдап алған дұрыс. Мысалы, күрделі алгоритмнің блок – схемасы мына суретте бейнеленген(Сурет 5).

Мұндай тәсілдердің бірі — бағдарламаны жинақталған ішкі бөліктерге бөліп, оларды кейін негізгі бағдарламаға біріктіру. Әдетте мұндай бөліктерді бағдарлама модулдері деп атайды. Бағдарламалаудың ыңғайлы тәсілдерінің бірі - модулдерді көмекші (ішкі) бағдарламалар құрастыру, сосын оларды орындалу ретіне сәйкес жоғарыдан төмен қарай орналастыру.

Алгоритмдерді құру әдістері. Алгоритмдерді құрудың көптеген әдістері бар, бірақ солардың ішінде танымал әдістерді атауға болады. Олар: *жеке бөлшектеу әдісі; өрлеу әдісі; кері шегіну арқылы құру әдісі; тармақтар мен шектеулер әдісі;*

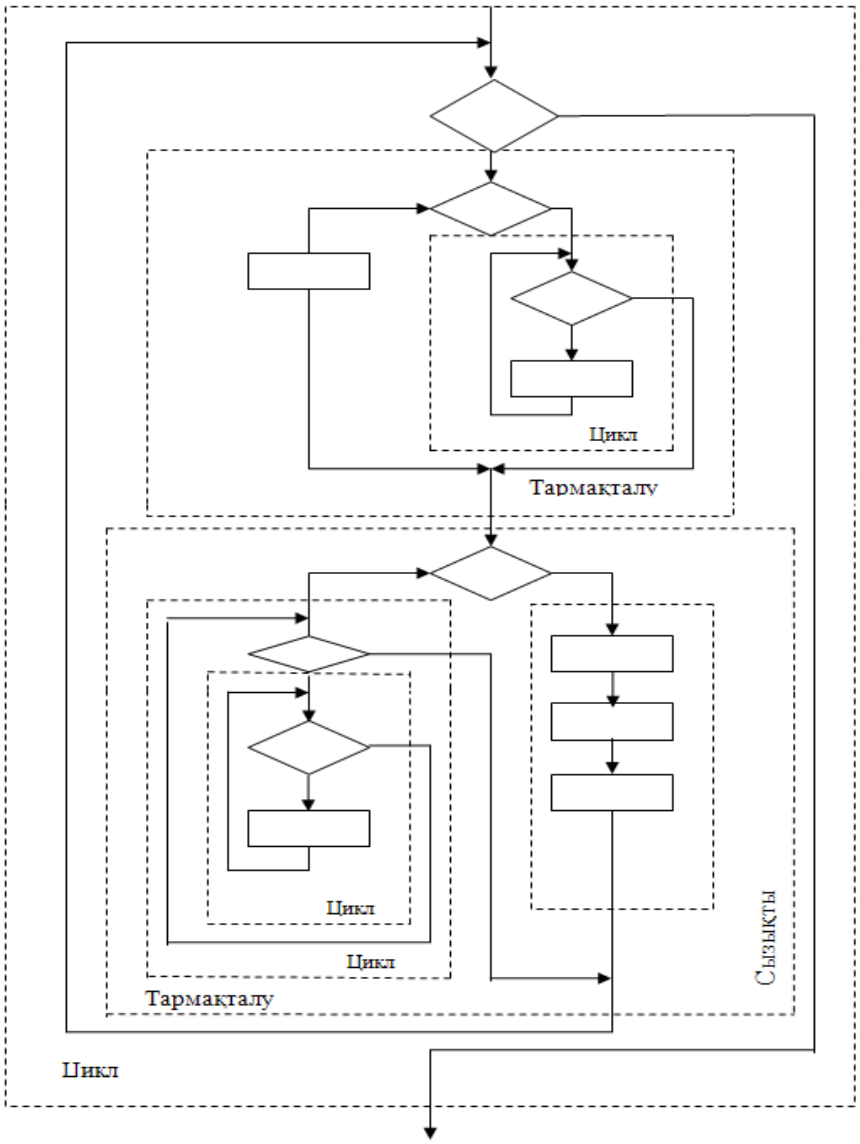
Жеке бөлшектеу әдісі. Бұл әдісті қысқаша былай тұжырымдауға болады: «Күрделі есепті жеңілдету үшін, оны ішкі есептердің тізбегіне келтіру қажет». Бұл әдісті әрбір қолданушы өз өмірінде жиі қолданады, әрбір алдына қойылған мақсатқа жету үшін біртіндеп жеңіл есептерді шеше отырып жетеді. Бірақ бұл жерде күрделі есепті жеңіл есептерге бөлу бағдарламалаушының тапқырлығын және тәжірибесін қажет етеді. Бұл әдісті қолданудың айқын мысалы - желілік жоспарлау есептері.

Өрлеу әдісі. Бұл әдістің негізгі мақсаты келесіде: алгоритм алғашқы болжам жасау және есептің алғашқы шешімін құрудан басталады. Содан кейін

алғашқы шешімнен мүмкін уақытқа дейін «жоғары» өрлейді. Ары қарай көтерілуге болмайтын, яғни алгоритм тоқтаған нүкте оның соңғы жоғарғы нүктесі болып табылады.

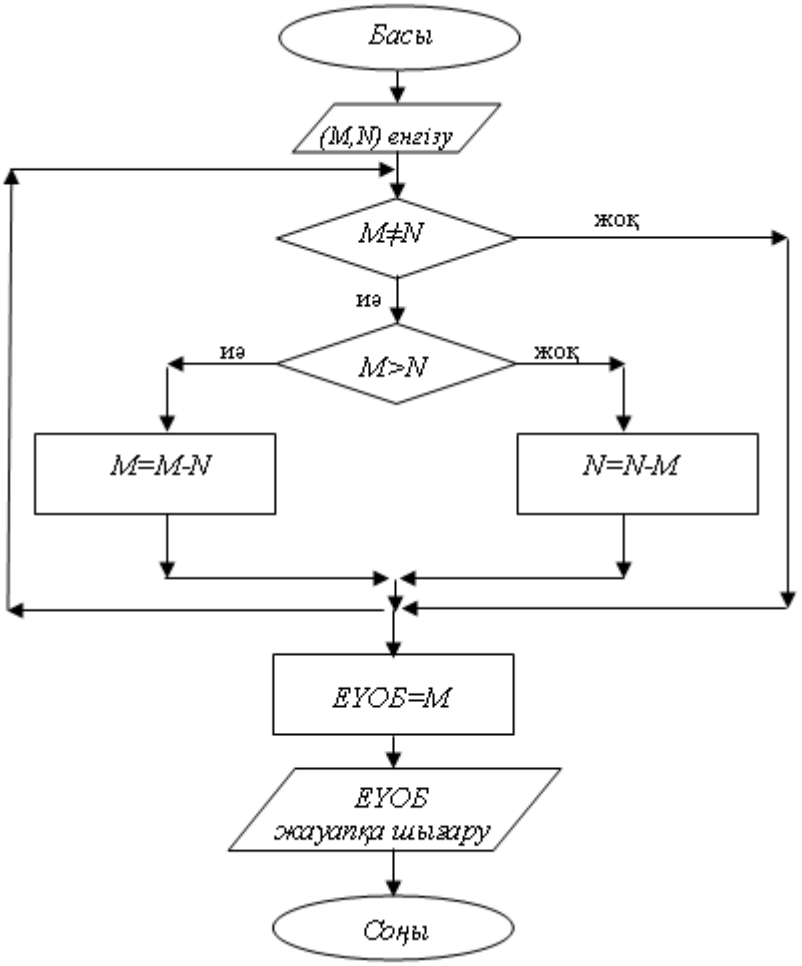
Кері шегіну арқылы құру әдісі. Бұл әдіс көбінесе іздеу есептерінде қолданылады. Ол ұйымдасқан толық іздеуді жүзеге асырады. Бұл әдісті қолданғанда көп вариантты шешімдерден құтылады. Мысалы, келесі типтес есептер үшін бұл әдіс жиі қолданылады: шахмат тақтасында кез келген екеуі бір вертикалда, горизонталда, диагоналда тұрмайтын 8 ферздің әр түрлі орналасуын ұйымдастыру керек.

Тармақтар мен шектеулер әдісі. Бұл әдіс саралау есептері үшін жиі қолданылады, яғни шешімдерді ағашқа ұқсас құрылым бойынша ең тиімдісін тапқанша жүргізіледі.



Сурет 5. Күрделі алгоритм схемасына мысал

Блок-схема түрінде жазылған алгоритмге мысал қарастырайық, *Екі бүтін санның ең үлкен ортақ бөлгішін (ЕҮОБ) табу керек.* Есептің алгоритмінің блок-схемасы(Сурет 6)



Сурет 6. ЕҮОБ есебінің блок - схемасы

Есепті шешуді оңайлату үшін рекурсивтік тәсілдер де пайдаланылады. Рекурсияның бір түрі - функция мәнін осы функцияның бұдан бұрын есептелген мәндері арқылы есептеу. Мұндай тәсілді рекуррентік тәсіл деп те атайды.

Мысал. *Cos x* функциясын функционалдық қатарға (шексіз қосындыға) жіктеу формуласы:

$$\cos x = 1 - x^2/2! + x^2/2! - \ldots = 1 + \sum_{k=1} (-1)^k \frac{x^{2k}}{(2k)!}$$

Қатардың eps-нен кем емес мүшелерінің қосындысын табу керек. Eps - есептеу дәлдігін білдіретін алдын ала берілген оң сан. Eps дәлдігі бойынша табылған мән осы дәлдікпен алынған cosx мәні де болатыны математика курсынан белгілі.

Нұсқау. Қатардың жалпы мүшесі (к=0, 1, 2,...):

$U = (-1)^k \cdot x^{2k} / (2 \cdot k)!$

Жалпы, берілген есепті бастапқы берілгендер мен шешуі бірдей болатын оңайлатылған ішкі есепке ауыстыру (ол бәсеңдету стратегиясы деп аталады), ал кезегімен оны да осы сияқты басқа ішкі есепке ауыстыру процесі рекурсия деп аталады.

1.2 Бағдарламалық қамтаманы жасау кезеңдері

Қазіргі коммерциялық бағдарламалық қаматаманы жасау бұрынғыларға қарағанда әлде-қайда қиынырақ. Бұл қиындықтың себебі бағдарламалық қамтаманы жобалаудың методологиясы бойынша жаңа зерттеулер жүргізіліп нәтижесінде, бағдарламаның құрылымына және жасау технологиясына қойылатын талаптар өзгертілді.

Бағдарламалық қамтаманы жасау технологиясына қойылатын негізгі талаптар: бағдарламалау тілдерінен тәуелсіз жұмыс істеу мүмкіндігі болу керек; қағазсыз технология негізінде, яғни бағдарламалық құжаттары да қосымша бағдарлама арқылы беріледі; бағдарлама жасаудағы құрастырушының ерекшелегі, оның стилі, дизайны және бағдарламалау әдісі, алгоритмі ерекше болу қажет.

Бағдарламалық жабдықтың өмірлік циклі(БЖӨЦ) – бағдарламалық өнімнің жобалаудан бастап жойылғанға дейінгі бар болу дәуірі. Циклдің негізгі кезеңдері: жобалау, дайындау, сынақтан өткізу, тәжірибелік пайдалану және сақтау, тұрақты пайдалану және істен шығару.

Бағдарламалық жабдықтарды жасау үшін алдымен пайдалануға қажетті деректер мен бағдарламалық құжаттар жүйесін және компьютерлерге арналған алгоритмдер мен бағдарламаларды әзірлеу қажет. Бағдарламаның өмірлік циклінің ұзақтығы бағдарламалық жабдықтың түрлеріне және қолданылуына байланысты болады.

Бағдарламалық өнімді даярлау қажеттілігін негіздеу: мәселенің қойылымын, алғашқы материалдарды жинау, әзірленетін бұйымның сапа шарттарын, қолдану тиімділігін тандау мен негіздеуді қамтитын техникалық тапсырма бойынша жасалатын жұмыстардың бірінші кезеңі.

Есептерді шешу алгоритмін анықтау: еңгізу-шығару және шешім қабылдау операциялары үшін стандартты графиқалық кескіндер жинағын пайдаланатын бағдарламаны әзірлеу тәсілі. Есептің алгоритмі көбінесе басқа әдістер (мыс., шешімдер бұтағы немесе шешімдер кестелері) арқылы дайындалады. Жүйелі талдауда қандай да бір үлкен жобаны әзірлеудің алғащқы сатысында пайдаланылады. Алгоритм бағдарлама жүзеге асыруға тиісті қадамдарды анықтау үшін қызмет етеді. Әдетте, есептің алгоритмі кіріс-шығыс деректерді, іске асырылатын әрбір алуан түрлі кіріс мәліметтерінің әрекеттерін, бағдарламаны бөлшектерге бөлуден басталады.

БЖӨЦ-не сәйкесті бағдарлама жасау кезеңдері: есептің қойылымы; математикалық моделін құру; алгоритмін құрастыру; бағдарламаны жазу; бағдарламаны жөңдеу және тестілеу; есептеулер жүргізу және алынған нәтижелерді талдау, бағдарламаны қайтадан тестілеу; бағдарламалық құжаттарын дайындау, бағдарламаны тарату және сүйемелдеу.

Есептің қойылымы: есеп туралы алғашқы мәліметтер жинау, есептің шартын қалыптастыру, соңғы нәтижелерді анықтау, деректерді сипаттау.

Алгоритмді құру: алгоритмдік құрылымның формасын таңдау, алгоритмді құру.

Бағдарламаны жазу: бағдарламалау тілін таңдау, деректерді көрсету формасын таңдау, бағдарламаның логикалық құрылымын құрастыру, тесттік тапсырмаларды және тестілеу әдісін таңдау, тестілеу және өңдеу, бағдарламаның жұмыс істеуін тексеру, тесттің нәтижелерін талдау, қателерді өңдеу, бағдарламаны қалыпқа келтіру, бағдарламаны орындау, қолданушы құжаттамасын дайындау

Бағдарламаны дайындау: бағдарлама атауы, автор туралы мәліметтер, бағдарламаны дайындаған мекеме, бағдарламаның дайындалған уақыты, мәселерді шешу тәсілдері, бағдарламаның орындалуына қажетті техникалық көрсеткіштер(жады,диск көлемі және т.б.), компьютер операторына нұсқау.

Бағдарламаны жөндеу. Синтаксистік қателер бағдарламаның компиляциялау поцесінде анықталады. Кейбір синтаксистік, сондай-ақ мағыналы немесе семантикалық қателер бағдарламаны тестілеу процесінде анықталады.

Сонымен, қарапайым есепті компьютерде орындау үшін мынандай кезеңдерден өту керек:

- *Есептің қойылымы:* есеп туралы ақпараттарды жинау; есептің шартын тұжырымдау; есептің мақсатын анықтау; деректерді сипаттау (олардың түрін, мәнін, құрылымын т.с.с.)
- *Есептің моделін талдау және зерттеу:* есептің құрылымын және техникалық және бағдарламалық құралдарды талдау; математикалық моделді және деректер құрылымын жасақтау.
- *Алгаритмді жасақтау:* алгоритмді жобалау тәсілдерін, алгоритмді жазу формасын (блок-схема, алгоритмдік тіл, т.б.), тесттерді және тестілеу тәсілдерін таңдау; алгоритмді жобалау.
- Бағдарлама*лау:* бағдарламалау тілін таңдау; деректерді ұйымдастыру тәсілдерін айқындау; алгоритмді бағдарламалау тілінде жазу.
- *Тестілеу және жөндеу:* синтаксистік жөндеу; семантикалық және логикалық құрлымын жөндеу; тестілік есептеулер және тестілеу нәтижелерін талдау; бағдарламаны жетілдіру.
- *Есептің нәтижесін талдау:* қажет болатын болса 2-5 кезеңдерді қайтадан орындай отырып, математикалық моделді айқындай түсу.
- *Бағдарламаны даярлау:* бағдарламаны жетілдіру; бағдарламаны пайдалану үшін құжаттарын жасау.

Математикалық моделдер

Зерттелетін объект немесе процесс олардың сандық параметрлерін байланыстыратын формула түрінде берілетін жағдайлар жиі кездеседі. Бұған мына төмендегі формулаларды жатқызуға болады:

- дененің геометриялық параметрлерін сипаттайтын формулалар;
- физикалық процестерді сипаттайтын формулалар;
- химиялық формулалар;
- тауардың бағасын есептеп шығаруға арналған тұрмыстық формулалар т.с.с.

Математикалық модел деп - объектіні немесе процесті, оларды сандық параметрлермен байланыстыратын математикалық формуламен сипаттауды айтамыз.

Математикалық модельдерді жазуда әр түрлі ғылымдарда қабылданған түрлі таңбалау жүйелері пайдаланылады. Оны жобалау көптеген есептерді компьютерде шығаруда маңызды рөл атқарады. Математикалық модель адамның шығармашылық жұмысының өнімі болып табылады. Адамның ойындағы жобаны таңбалық пішінге аудару. Мысалы ретінде дененің түзу сызықты орын алмастыру формуласын алуға болады:

$$X = X_0 + V_{xt} + \frac{a_{xt^2}}{2}$$

мұнда, х- ағымдағы координата, x_0- бастапқы координата, v_x- бастапқы жылдамдықтың х осіне проекциясы, a_x- үдеудің х осіне проекциясы, t- уақыт.

Құбылысты сипаттау үшін оның ең маңызды қасиеттерін, заңдылықтарын, ішкі байланыстарын, оның кейбір жеке сипаттамаларын бөліп көрсетіп, маңызды еместерін ескермеуге болады. Құбылыстың метематикалық моделін жасақтауда осы айтылғандарды ескеру қажет.

Сонымен, метематикалық моделін жасақтауда мыналарды ескеруіміз керек:

- математикалық модел құруда негізге алынатын болжам жасау;
- алғашқы деректер және нәтижелерді анықтау;
- алғашқы деректер мен нәтижелер арасындағы байланысты математикаға қатысты жазу.

Математикалық моделді құруда алғашқы деректер арқылы іздеп отырған шаманы өрнектейтін формуланы табу барлық уақытта мүмкін бола бермейді. Мұндай жағдайда белгілі бір немесе басқа дәрежелі дәлдікпен жауап беруге мүмкіндік беретін математикалық әдістер пайдаланады.

Ғылыми есептерді шығарғанда мүмкіндігінше шындыққа жақын келетін математикалық модел құру керек болады. Мұндай модел бойынша дәл есептеулер емес жуықтап есептеулер жасалады, бірақ осының арқасында басқа жолдармен алуға болмайтын деректерді алуға болады. Есеп математикалық моделге келтірілгеннен кейін оны шығарудың алгоритмін құруға болады. Алгоритмнің дайын болуының белгілі бір кезеңінде бағдарламалау басталады.

Өмірде күрделі есептерді шығаруда үлкен ұжымдар жұмыс істейді. Есептің бірінші формалды түсінігін сол есеп пайда болған саланың маманы даярлайды. Математикалық моделді басқа адамдар жасай алады, ал ондағы пайдаланылатын алгоритмдерді үшінші бір адамдар жасайды. Ең соңында бағдарламалауды тағы бір мамандар тобы іс жүзіне асыра алады. Осылайша, есеп үлкен ұжымның бірлескен еңбегінің арқасында шығарылатын жағдайлар жиі кездеседі.

1.3 Бағдарламаны жөндеу және тестілеу

Бағдарламаны жөндеу (debugging- жөндеу ағылшын сөзі, сөзбе - сөз аударғанда *қоңыздарды аулау* деген мағына береді.) дегеніміз бағдарламаны компьютерде орынлау нәтижесі бойынша ондағы қателіктерді іздеу және жою процесі.

Тестілеу (test ағылшын сөзі *сынау* деген мағына береді) дегеніміз бағдарламаның немесе оның жеке бөліктерінің дұрыс жұмыс істейтіндігін сынау, тексеру.

Жөндеу және тестілеу бұлар бірінен бірі айқын ажыратылған және біріне-бірі ұқсамайтын екі кезең:

- жөндеу кезінде синтаксистік қателіктердің және кодтаудың айқын қателіктерін қадағалау және жою орындалады.
- тестілеу процесі кезінде айқын қателіктері жоқ бағдарламаның жұмыс істеу қабілеттілігі тексеріледі.

Тестілеу қателіктің бар екендігін тағайындайды, ал жөндеу бағдарламаның дұрыс жұмыс істемеу себептерін анықтайды.

Қазіргі заманғы бағдарламаны жөндеу жұмысы жөндеуші деп аталатын арнайы бағдарламалық құралды пайдалану жолымен жүзеге асырылады. Бұл құралдар бағдарламаның ішкі жұмысын зерттеуге мүмкіндік береді.

Жөндеу бағдарламасының мына төмендегідей мүмкіндіктері бар:

- бағдарламаның әрбір командасын қадамдап орындау;
- кез келген айнымалының ағымдағы мәнін қарау немесе кез келген өрнектің, оның ішінде стандартты функцияларды пайдаланып мәнін табу қажеттілігіне қарай айнымалының жаңа мәнін пайдаланып орындау;
- бағдарламада «тексеру нүктелерін» орнату, яғни аралық нәтижелерді бағалау үшін бағдарлама уақытша өз жұмысын тоқтататын нүктелерді тағайындау т.б.

Бағдарламаны жөндеуде мыналарды есте сақтау маңызды:

- жөндеу процесінің басында қарапайым тестілік деректерді пайдалану;
- пайда болатын қиындықтарды айқын ажыратып және қатаң түрде кезегімен жою.

Тест дегеніміз - алғашқы деректердің кейбір жиыны және осы деректерге сәйкес келетін бағдарлама жұмысы кезінде алынуға тиісті барлық нәтижелерінің дәл сипаттамасы.

Бағдарлама қаншалықты өте мұқият жөнделсе де оның жұмысқа жарамдылығын тағайындаудың шешуші кезеңі бағдарламаның тесттер жүйесінде тікелей орындалуын тексеру болып табылады. Егер бағдарлама таңдалынып алынған тесттілік деректер жүйесінен өткенде барлық жағдайда дұрыс нәтижелер беретін болса, онда бағдарламаны шартты түрде дұрыс деп есептеуге болады.

Тестілеу тәсілін жүзеге асыру үшін тесттердің эталондық нәтижелері алдын-ала даярланған немесе белгілі болуы тиіс. Эталондық нәтижелерді бағдарламаның компьютерде орындалу нәтижесін алғаннан кейін емес, керісінше алғанға дейін есептеп шығарып қою керек.

Тестілік деректер қателіктердің пайда болуының барлық мүмкін болатын шарттарын тексеруді қамтамасыз ететін болуы тиіс, атап айтқанда: алгоритмнің әрбір тармағын сыннан өтуі; бағдарламаның жалпы жұмысын тексеру үшін бірінші тестің мүмкіндігінше қарапайым болуы; есептеу көлемін қысқарту үшін тестідегі арифметикалық операцияларды ықшамдау.

Тестілеу процесін екі түрлі жағдайда қарастыруға болады.

1. *Қалыпты жағдайда тексеру.* Бағдарламаның жұмыс істеуінің нақты жағдайына тән алғашқы деректер негізінде тестілеу.

2. *Экстремальды жағдайда тексеру.* Бұл жерде тестілік деректер дегеніміз алғашқы деректер жиынының шекаралық мәндері. Бұл жиын бағдарлама дұрыс жұмыс істеуге тиісті алғашқы деректерден тұрады. Мысалы, өте кіші немесе өте үлкен сандар және деректердің болмауы қарастырылады.

Жаңадан ғана жасалған бағдарлама қателіктің болуы - бұл қалыпты және заңды құбылыс. Іс-жүзінде нақты және жеткілікті дәрежеде күрделі програмаларды қателіксіз құру мүмкін емес.

Бағдарламаны компьютердің қабылдауына және оның нәтижесіне қарап оны дұрыс құрылған деген қорытынды жасауға болмайды. Өйткені берілген жағдайда дұрыс болмаса да қандай да бір нәтиже алуға қолымыз жетті. Осының өзінде бағдарламада көптеген логикалық қателіктердің болуы мүмкін. Кеткен қателіктерді анықтау үшін бағдарламаны компьютерге ендіріп, іске қосқанға дейін былай тексереміз. Бағдарлама мәтінін «қолмен» қарап шығу, тексеру және айналдыру арқылы тексеруге болады.

Бағдарлама мәтінін қарап шығу. Бағдарлама мәтінін жазуда кеткен қателіктерді және бағдарламаның алгоритмнен алшақтығын анықтау үшін қарап шығады. Атап айтқанда итерация санын беуші операторлардың дұрыстығына көз жеткізу үшін барлық циклдердің ұйымдастырылуын қарап шығу өте маңызды. Шартты операторлардағы шарттарды, бағыныңқы бағдарламаларға қатынас жасаудағы аргументтерді тексеріп шығу пайдалы.

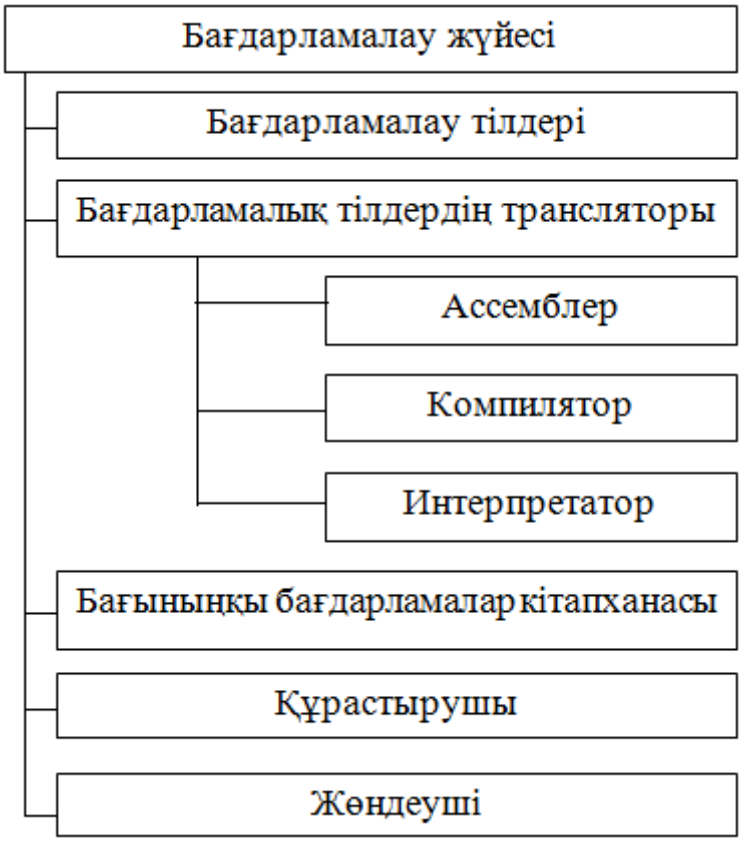
Бағдарламаны тексеру. Бағдарламалаушы бағдарламаны тексергенде оның мәтіні бойынша бағдарлама анықтайтын есептелу процесін ойша қалпына келтіруге тырысады, осыдан кейін оны талап етілетін процеспен салыстырады.

Бағдарламаны айналдыру. Айналдырудың мәні бағдарламаның компьютерде орындалуын бағдарламалаушының ойша орындауы болып табылады. Айналдыруды орындау үшін кейбір алғашқы деректерді беріп және олармен қажетті есептеулерді жүргізу қажет. Айналдыру - бұл еңбекті көп талап ететін қиын процесс, сондықтан оны тек бағдарламаның логикалық күрделі бөліктеріне қолданған жөн. Алғашқы деректерді бағдарламаны айналдыру кезінде бағдарламаның зерттелетін көптеген тармақтары қамтылатындай етіп таңдап алу керек.

Бағдарламалауда кездесетін қателіктер туралы толық мәліметті мына оқу –әдістемелік құралынан көруге болады[3]

2 Бағдарламалау жүйесі

Бағдарламалау жүйесі компьютердің бағдарламалық қамтамасының құрамына жатады және ол бағдарламаларды жасау және оларды жөндеу жұмыстарын автоматтандыруды қамтамасыз ететін құралдардың жиыны болып табылады. Бағдарламалау жүйесінің құрамы мына төмендегі схемада бейнеленген (Сурет 7).



Сурет 7. Бағдарламалау жүйесінің құрамы

Бағдарламалау процесі үш кезеңге бөлінеді:

- есепті шешудің алгоритмін құру;
- бағдарлама құру;
- жасалған бағдарламаны тексеру.

Екінші кезеңдегі, яғни бағдарлама құрудағы қиындық адамның тек машина тілінде ғана бағдарлама жасауына байланысты болады. Компьютерді пайдаланудың алғашқы жылдарында әр түрлі компьютерлердің пайда болуы бұл қиындықты тереңдетіп жібереді. Сондықтан нәтижесінде бір компьютерге арнап жасаған бағдарламаны басқа компьютерге пайдалануға болмайтын болды. Бұл бір алгоритмнің өзін әр түрлі компьютерлерде орындау үшін әрқайсына жеке-жеке бағдарлама құруға мәжбүр етіп жұмысты қиындатып жіберді.

Әрбір компьютер үшін жасалған бағдарламаның дұрыс және ұтымды жасалғандығын тексеру қажет. Қателік кеткендігі және басқа да кемістіктері

бар екендігі анықталса, онда ол бағдарламаға түзетулер мен өзгерістер ендіріледі. Бағдарламалаудың осы аталған кезеңдерінің арнайы ерекшеліктері және қиыншылықтары бар.

Барлық осы аталған қиыншылықтар бағдарламалар жасаудағы атқарылатын жұмыстарды мейлінше ықшамдап қысқарту мәселесін күн тәртібіне қояды. Бұл мәселені шешу бағдарламалау процесін автоматтандыруды, яғни компьютердің өзін бағдарламалар жасауға немесе бағдарламалауға байланысты таза техникалық жұмыстар атқаруға пайдалануды талап етеді. Автоматтандыру бағдарлама жасаушы мамандардың жұмысын жеңілдетуді, оны жасауда кететін қателіктердің санын азайтуды, ең соңында есептің қойылуынан бастап нәтиже алғанға дейін кететін уақытты қысқартуды көздейді. Бағдарламалауды автоматтандырудың ең көп тараған негізгі тәсілдеріне мыналар жатады:

– қолмен бағдарламалауды жеке жұмыстарын автоматтандыру тәсілдері;
– бағыныңқы бағдарламалар кітапханасын құру;
– бағдарламалаудың әр түрлі тілдерін пайдалану.

Қолмен бағдарламалаудың жеке жұмыстарын автоматтандыру тәсілдері. Қолмен бағдарламалаудағы жұмыстарды автоматтандыруда жұмыс бағдарламаларын түзетуді негізгі назарда ұстау керек.

Қазіргі кезде бағдарламаның дұрыстығын тексеру үшін жоғарыда айтқандай тест тәсілі кеңінен қолданылады. Тест тәсілі бойынша жауаптарды алдын ала белгілі есептердің бірнеше нұсқасы компьютерде құрылған бағдарламаны пайдаланып шығарылады. Алынған нәтижелер алдын ала белгілі жауаптармен салыстырылады. Ол жауаптар біріне – бірі сәйкес келмесе, онда бағдарламада қателік кеткен болып шығады.

Бағдарламаны жөндеу процесі бірнеше әмбебеп жөндеуші бағдарламаларды жасау және компьютердің өзін пайдалану арқылы автоматтандырылады. Бұл бағдарламалар командалардың қалай орындалып жатқандығы туралы ақпаратты баспаға бере отырып жөнделіп жатқан бағдарламаның кез келген бөлігінің жұмысын қадағалап бақылауға мүмкіндік береді. Бағдарламаны автоматты түрде жөндеуде бағдарламалаушыны қызықтыратын барлық ақпарат құжатқа жазылады, соның арқасында ол компьютерді пайдаланбай-ақ есептің компьютерде шешілу процесінің дұрыстығын бақылай алады.

2.1 Бағдарламалау тілдері

Бағдарламаларды жасауда қазіргі заманғы компьютерлерде әр түрлі деңгейдегі бағдарламалау тілдері пайдаланылады. Жалпы жағдайда *тіл* деп ақпараттың жазылуын және оның түрленуін белгілі ережелер бойынша формалды етіп беруші құралдарды түсінеміз.

Тілдің құрамында ақпараттың негізгі элементтерін құратын құралдары және осы элементтерден әр түрлі мазмұнды құрылымдар құруға арналған ережелер жүйесі бар.

Бағдарламалау тілі деп - деректерді жазуға және оларды белгілі ережелер бойынша өңдеуге арналған адам мен компьютерді байланыстыратын формалды тілді айтамыз.

Әр түрлі белгілер бойынша жіктеуге болатын бірнеше жүздеген бағдарламалау тілдері бар. Ең жалпысы тілдің машинаға жақындық дәрежесі бойынша жіктеу болып табылады. Осы белгісі бойынша бағдарламалау тілдері екі үлкен топқа бөлінеді: *машинаға тәуелді тілдер; машинаға тәуелсіз тілдер.*

Машинаға тәуелді тілдер өз кезегінде былай бөлінеді: *машина тілі; машинаға бейімделген тілдер.*

Машинаға бейімделген тілдер кейде автокодтар деп те аталады. Машинаға бейімделген тілдердің екі деңгейі бар: *символдық кодтау тілдері, басқаша айтқанда мнемокодтар; макротілдер.*

Макротілдер машина тілінің командаларына тікелей ұқсастығы жоқ *макрокомандаларды* пайдалануға рұхсат етеді. Макрокомандаларды пайдалану бағдарламаны қысқартады әрі тілді жасау құралдарының жиынын кеңейте отырып бағдарламалаушының еңбегінің өнімділігін арттырады.

Машинаға тәуелсіз тілдер бағдарламаларды бөлшектеу дәрежесіне қарай екі топқа бөлінеді: *процедуралы- бейімделген тілдер; мәселелік – бейімделген тілдер.*

Процедулалы – бейімделген тілдер есепті шешу алгоритмін сипаттауға арналған, сондықтан да оларды кейде *алгоритмдік тілдер* деп те атайды.

Мәселелі – бейімделген тілдер есептерді сипаттау үшін қызмет атқарады.

Бірақ алгоритмдік тілдер өздігінен барлық мәселені, тіптен бағдарламаны да шеше алмайды. Мұндай тілде жазылған бағдарлама компьютерде тікелей орындалмайды, ол орындалу үшін алдын ала сол компьютердің машина тіліне аудару жеткілікті қиын мәселе болып табылады.

Алгоритмдік тілден бағдарламаны машина тіліне аудару жұмысын компьютердің әмбебаптығын пайдаланып, оның өзіне жүктеуге болады. Бұл үшін әрбір алгоритмдік тілден машина тіліне формалды аударуға яғни аударудың ережесін алгоритм түрінде тұжырымдауға мүмкіндік беретіндей болуы керек. Егер, осы алгоритмді бір рет машиналық бағдарлама түріне келтіріп алсақ, онда одан кейін осы бағдарламаның көмегімен компьютердің өзі берілген алгоритмдік тілде жазылған кез келген бағдарламаны нақтылы машина тіліне аударып бере алады. Мұндай арнайы аудармашы-бағдарлама *транслятор* деп аталады.

Компьютердің ақпараттық бөлігін тікелей түсінетін жалғыз тіл: ол – машина тілі. *Машина тілі* деп копьютердің құрамындағы процессор командаларының кодын айтамыз.

Архитектурасы әртүрлі процессорлардың машина тілдері де түрліше болады. Тек өзара үйлестірілген процессорларда ғана машина тілдері бірдей болады. Мұндай процессорлардың командалар жүйесі төменнен жоғары қарай үйлесімділікте болады.

Сонымен әрбір компьютердің өзінің *машина тілі* болады және ол тек осы тілде жазылған бағдарламаларды ғана тікелей орындай алады.

Машина тілінде бағдарламалау деп бағдарламаға енетін командалардың нақты кодтарын тікелей жазуды айтамыз. Командалардың кодтары әр түрлі санау жүйелерінде берілуі мүмкін: екілік; сегіздік; он алтылық.

Машина кодтарын пайдаланып бағдарламалау үшін қарапайым жағдайда қағаз, қалам және ақпаратты тасымалдаушыға машина кодындағы бағдарламаны жазуға арналған перфоратор болса жеткілікті. Командалардың он алтылық кодтары пернетақтадан компьютердің жадына ендіріледі, *монитор* бағдарламаны түзету процесін басқарады, ал экран командалардың коды мен деректерді он алтылық кодта бейнелеп көрсетеді. Машина кодын пайдаланып бағдарламалауда еңбек өнімділігін артыру үшін командалардың мнемоникасын және коды берілген анықтамалық кестелерді пайдалануға болады.

Бірақ қазіргі кезде машина тілінде бағдарламалау тек қысқа бағдарламаларға шамалы өзгерістер өндіру қажет болған жағдайларда ғана қолданылады, өйткені бұл бағдарлама жасаушылар кездесетін мынадай қиыншылықтарға байланысты:

- процессор командаларының көптеген кодтарын есте сақтауға тура келеді;
- жадтың абсолюттік адрестері, әсіресе шартты өтулердің саны көп, сондықтан ұзын бағдарламалар жасауда қадағалау өте қиын;
- жазылған бағдарламаны қайта жетілдіріп, өзгертіп жасау өте күрделі жұмыс болады;
- машина кодтарында жазылған бағдарламалар өте қиын оқылады;
- тек қана сандардан тұратын бағдарламаны жасау бағдарламаны жасаушыны жалықтырып жібереді және бағдарламаларда қателіктердің келуіне алып келеді.

Осылайша машина тілі цифрлар тілі бола отырып, бағдарлама жасаушы адамнан бағдарламаны жазуға және оны жөндеуге көп уақыт жұмсауды талап ететіндіктен, бағдарламалау үшін жарамдылығы шамалы болады.

Бағдарламалаудағы қиыншылықтарды азайтуға бағытталған келесі жоғары деңгей тілдер пайда болды:

- 1955 ж. алғашқы жоғары деңгейлі FORTRAN (FORMULA ANSLATOR) – ғылыми есептеулер тілі пайда болды.
- 1960 ж. Алгол (Algoritmic Language) пайда болды.
- 1965 ж. Basic (Beginner’s All purpose Symbolic Instructions Code) пайда болды. Авторлары Курт және Кемени.
- 1970 ж. Никлаус Вирт студенттерге бағдарламалауды үйретуге арналған Паскаль тілін шығарды.
- 1972 ж. Си тілі пайда болды (авторы Денис Ричи).
- 1980 ж. 3-ші буындағы ең күшті Ада тілі пайда болды.

Жоғары деңгей тілдерінің пайда болуымен бағдарламаларды құру жылдамдығы жоғарылады, бағдарлама құрудағы уақыттың 80%-н алатын түзету процесін автоматтандыру мүмкіндігі пайда болды[4].

Ассемблер және макроассемблер тілдері

Ассемблер таңбалар (символдар)тілі бола отырып, белгілі бір дәрежеде машина тілінде бағдарлама жасаудағы кемшіліктерді жоюға мүмкіндік береді.

Ассемблер тілінде бағдарламаның барлық элементтері таңбалармен берілетіндігі оның басты артықшылығы болып табылады. Басқаша айтқанда ассемблер тілінің машина командаларының цифрлық кодтарын әріптермен немесе әріп-цифрлармен таңбалауға және деректердің таңбалық аттарын пайдалануға мүмкіндік беретіндігі оның машина тілінен айырмашылығы болып табылады. Ассемблер тілінің командаларын машина тіліне аударғанда машина командасын білдіретін әрбір оператор осы команданың цифрлық кодтарымен алмастырылады. Командалардың таңбалық аттарын олардың екілік кодтарына түрлендіру жұмысы бағдарлама жасаушы адамды өте қиын әрі күрделі машақаты көп жұмыстан босататын және бұл жағдайда құтылуға болмайтын қателіктерден құтқаратын арнайы *бағдарлама-ассемблерге* жүктеледі.

Ассемблер тілінде бағдарламалауда пайдаланылатын таңбалық атаулар бағдарламаның семантикасын, ал команданың қысқартылып берілген атаулары оның негізгі функциясын білдіреді. Мысалы, ADD-қосу, SUB-азайту, PARAM-параметр т.с.с. Мұндай аттарды бағдарлама жасаушылар оңай есінде сақтайтын болады.

Ассемблер тілінде бағдарлама жасау үшін машина тілінде бағдарлама жасағандағыдан көп күрделі құралдар қажет болады:

- сыртқы құрылғылармен жабдықталған дербес компьютер;
- процессордың түріне қарай резиденттік немесе жүйелік бағдарламалар.

Машина тіліне қарағанда ассемблер тілі едәуір күрделі, бағдарламаларды ұтымды жазуға және жөндеуге мүмкіндік береді.

Ассемблер тілі машинаға бағдарланған тіл, яғни процессордың әрбір командасына таңбалық ат меншіктейтіндіктен машина тіліне және процессордың құрылысына тәуелді тіл болып табылады.

Бағдарлама жасауда ассемблер тілі машина тіліне қарағанда бағдарлама жасаушылардың еңбек өнімділігін арттыруға сонымен бірге процессордың бағдарламалық және аппараттық ресурстарын толық пайдалануға мүмкіндік береді.

Ассемблер тілінің осындай артықшылықтарына байланысты ендіру-шығару құрылғыларын басқарушы бағдарламалар неше түрлі жоғары деңгейлі тілдердің көптігіне қарамай ассемблер тілінде жазылады. Ассамблер тілінің көмегімен бағдарлама жасаушы адам мынадай параметрлерді бере алады:

- процессордың машина тілінің әрбір командасының таңбалық атын;
- ассемблер тілінде жазылған бағдарламалық қатардың стандартты пішімін;
- командалардың нұсқаларын және адрестеудің тәсілдерін қалай көрсету керектігі туралы пішімді;
- таңбалық тұрақтыларды және бүтін сандық тұрақтыларды әртүрлі санақ жүйелерінде көрсету үлгілерін;
- бағдарламады ассемблерлеу(транслациялау) процесін басқарушы пседокомандаларды.

Ассемблер тілі кез келген компьютерге түсінікті, өйткені басқа барлық тілдерге қарағанда машина тіліне ең жақын тіл - ол ассамблер тілі. Бұл тіл копьютермен жақынырақ танысуға мүмкіндік береді. Сондықтан да ассемблерді оқу дегеніміз процессордың өзін оқып үйрену деген сөз. Ассемблер тілінде жазылған бағдарлама кез келген басқа тілдерде жазылған бағдарламаларға қарағанда өте тез орындалады. Мысалы, ассемблер тілінде жазылған бағдарлама дәл осы бағдарламаға баламалы СИ немесе Паскаль тілдерінде жазылған бағдарламалардан екі-үш есе, ал BASIC тілінде жазылған бағдарламалардан он бес және онда да көп есе тез орындалады.

Ассемблер тіліндегі бағдарламалар басқа тілдерде жазылған бағдарламаларға қарағанда өлшемі жағынан шағын болады, сондықтан компьютердің жадын үнемдеуге мүмкіндік береді. Ассемблер тіліндегі бағдарламалар компьютердің барлық мүмкіндіктерін толық ұтымды пайдалануға сізге жол ашады.

Microsoft NET платформасы

Microsoft NET платформасы қуатты интеграцияланған IDE жұмыс істеу ортасына негізделген. Барлық .NET тілдеріне ортақ CLR (Common Language Runtime) орындаушылық ортасы бар. CLR ортасында орындалуға арналған кез-келген тілде жазылған код, CLR спецификациясына сәйкес келетін кез-келген тілде орындала алады. Мысалы, VB коды C# тілінде жазылған бағдарламаларда қолданыла алады және керісінше.

Сонымен қоса, мұнда бағдарламалаушы тарапынан ешқандай күш жұмсалмайды. Орындалушы кодтың жалпы форматы, аралық тіл деп аталатын Microsoft MSIL (Microsoft Intermediate Language) жасалды. Кез-келген тілде жазылған бағдарлама MSIL–кодқа компиляцияланады, бұл тіларалық сәйкестікті қамтамасыздандырады. MSIL коды .NET ортасында орындалуы кезінде машиналық кодқа өрнектеледі.

NET платформасының артықшылықтары:

- Бағдарламалар құруды жеңілдететін бағдарламалаудың объектіге-бейімделген моделі.
- Барлық тілдерге ортақ CLR орындаушы ортасы.
- Барлық .NET тілдері үшін MSIL орындалушылық кодының жалпы форматы.
- Көпағындылық. Бір бағдарламаның бірнеше қызметтін бір уақытта орындау.
- Кодты көбейту мен сүйемелдеу қауіпсіздігінің қазіргі үлгісі.
- Реестрден толық бас тарту.
- Қауіпсіз типтер және қосымшалар қауіпсіздігінің жалпы артуы.
- Қателерді өңдеудің бірегей үлгісі.
- Кодты қайта қолданудың кеңейтілген мүмкіндіктері.

NET платформасының кемшіліктері:

- Бағдарламалардың жай орындалуы.
- Кейбір .NET архитектуралық шешімдердің C++ тәріздес тілдерге байланыстылығы.

2.2 Бағдарламалық тілдердің трансляторлары

Қазіргі кезде компьютерлік технологиялар әлемінде әртүрлі типтегі машиналарға арналған бағдарламалау тілдері көптеп таралуда. Жоғары деңгейлі тілдер көптеген алгоритмдерді үйреншікті математикалық амалдардың жазылуына жақын, ыңғайлы түрде жазып түсіндіруге мүмкіндік береді. Бұл тілдерді пайдалану бағдарламалаудағы кездесетін қиындықтарды азайтады.

Жоғары деңгейлі тілдерді бағдарлама жасауда пайдалану 60-шы жылдары басталды. Содан бері бүгінгі күнге дейін белгілі есептерді шешуге арналған әмбебап, сандай-ақ бейімделген көптеген әртүрлі тілдер жасалып пайдаланылып келеді[5].

Әрбір бағдарламалау тілінің өзінің аты бар. Көптеген бағдарламалау тілінің аты олар алғаш жасалғаннан бастап тіркелген. Содан бері бағдарламалау тілдерінде қолданылатын ережелер өзгергенімен тілдерің аты сол бұрынғы күйінде өзгеріссіз қалуда. Қазіргі кезде жоғары деңгейлі бағдарламалау тілдері былай топталады:

- процедуралы (классикалық бағдарламалау тілдері, мысалы, FORTAN, PASCAL, BASIC, C);
- логикалық (ЛИСПЫ, ПРОЛОГ т.б.);
- объектіге - бейімделген (C^{++}, Java т.б.).

Қысқа бағдарламаларды жасауда процедуралық бағдарламалау тілдерін пайдалану ыңғайлы; логикалық бағдарламалау тілдерін алгоритмдерді теориялық зерттеуде, жасанды интеллекті оқытып үйрену жұмыстарында, деректер базасымен жасалатын операцияларда өндіріс объектілерін және әскери бөлімдерді басқару жүйелерін басқаруда, ал объектік- бейімделген бағдарламалау тілдерін бәрінен де үлкен және күрделі бағдарламаларды (Мысалы, компьютерлік ойындарда), жасақтауда пайдаланған жөн.

Әртүрлі бағдарламалау тілдерінің арасында айтарлықтай елеулі айырмашылықтардың болуына қарамастан олардың барлығында негізгі операцияларды жүзеге асырудың ұқсас құралдары бар. Бұдан басқа әртүрлі бағдарламалау тілдерін пайдаланғанда жұмыс істеу қиындығының түрліше болатындағына қарамастан кез келген бағдарламаны жасауда кез келген типтегі бағдарламалау тілін пайдалануға болады.

Бұл тілдердің әрқайсысына тоқталмай-ақ, осы тілдерді машина тіліне аударушы трансляторлардың жұмысына тоқталайық. Машина тілінен өзгеше бағдарламалау тілінде құрылған бағдарлама компьютерде орындалу үшін жарамды түрде, яғни машина тіліне түрлендіруі тиіс. Мұндай түрлендіру трансляциялау деп аталады.

Ассемблердің ендірілетін тілі мнемокод, макроассемблер-макротіл, ал компилятордікі - поцедуралы бейімделген тілдер болып табылады. Осыған байланысты ендірілетін тілдерді транслятордың түрлеріне қарай ассемблер тілі, макроассемблер тілі деп аталады т.с.с.

Транслятор арқылы өңделіп алынған бағдарлама тікелей компьютерде орындалады немесе оны басқа транслятордың өңдеуіне тура келеді.

Трансляциялау мен бағдарламаның орындалуы уақыт жағынан бөлінген болады. Интерпретатордан басқа трансляторларда алдымен барлық бағдарлама трансляцияланады содан кейін орындалады. Осы режімде жұмыс істейтін трансляторлар компиляциялаушы типті трансляторлар деп аталады. Егер мұндай транслятордың ендірілетін тілі процедуралы- бейімделген тіл болса, онда транслятор *компилятор* деп аталады.

Трансляциялау кезінде орындау кезеңдері уақыт бойынша ығысып ауысып келіп отыратын транслятор *интерпретатор* деп аталады. Машина тілінде немесе жүктелуші тілде ұсынылған бағдарлама транслятор жұмысының нәтижесі болып табылады.

Транслятордың жұмысын төрт кезеңге бөлуге болады:

– *лексикалық талдау*. Мұның негізгі атқаратын қызыметі бағдарламаның бастапқы мәтінін одан әрі қарай өңдеу үшін ең ықшамды және ыңғайлы етіп ұсыну. Осылайша алынған мәтін транслятордың синтаксистік талдаушы деп аталатын келесі бөліміне бастапқы деректер ретінде беріледі;

– *синтаксистік талдау.* Бұл кезңде бастапқы мәтінді синтаксистік талқылау жүргізіледі, яғни сөйлемдердің типтерін тану және бағдарламаның құрылымын айқындау, сонымен бірге синтаксистік қателіктерді айқындаушы синтаксистік бақылау;

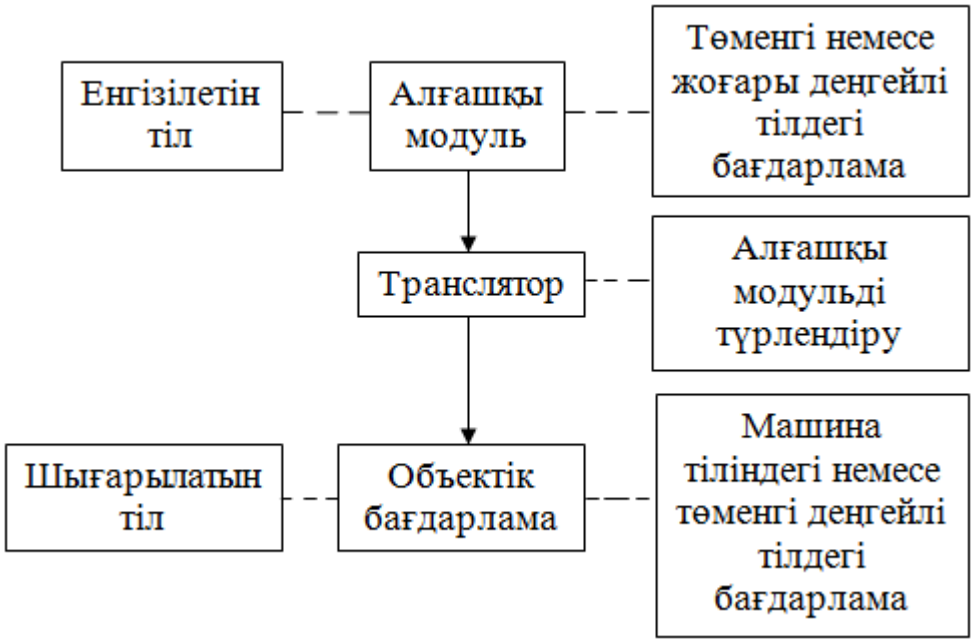
– *объктік бағдарламаны жасау.* Бұл кезеңде жасанды тілдің баламалы сөйлемдерінің мәні зерттеліп, симантикалық талдау жасалады;

– *объктік бағдарламаны безендіру және беру.* Бұл транслятор жұмысының соңғы қорытынды кезеңі. Объктік бағдарламаны кітапханаға жазуға, баспаға шығаруға болады. Пайдаланушының нұсқауы бойынша транслятор ендіретін қосымша ақпараттың белгілі бір бөлігі ғана баспаға беріледі.

Шығарылатын есептің сипатына және пайдаланушылардың категориясына байланысты трансляторларға әртүрлі талаптар қойылады. Мысалы, берілген бағдарламалау тілін игергісі келетін жаңадан бастаушыларға транслятордың ең маңызды сипаттамасы диагностикалық хабарларының толық әрі қарапайым болуы болып табылады. Егер компьютерде салыстырмалы түрде алғанда көп уақыт есептеуді қажет етпейтін көптеген майда есептер шығарылатын болса, онда сол есептерді шығару үшін алынған бағдарламаның сапасына айтарлықтай мән берілмейді.

Транслятордың жұмыс істеу жылдамдығы үлкен рол атқарады. Ұзақ есептелетін күрлелі есептер үшін талап етілетін машиналық уақытты және бағдарламаның орындалуы үшін қажетті жадтың көлемін ескере отырып транслятордың жасаған бағдарламаның ұтымдылығы ең маңызды рол атқарады. Мұндай ұтымды бағдарламалар алу трансляциялау алгоритмдерін күрделендіре түсуді талап етеді, ол трансляторды күрделендіре түсуге және оның жұмыс істеу уақытын арттыруға алып келеді. Осыған байланысты бағдарламалау жүйесінің өзінде тіптен бір бағдарламалау тілі үшін де бірнеше әртүрлі трансляторлар қарастырылады, ал пайдаланушы өзіне ең керекті

трансяторларды таңдап алады. Трансляциялаудың жалпы схемасы(Сурет 8) мына төменде көрсетілген.

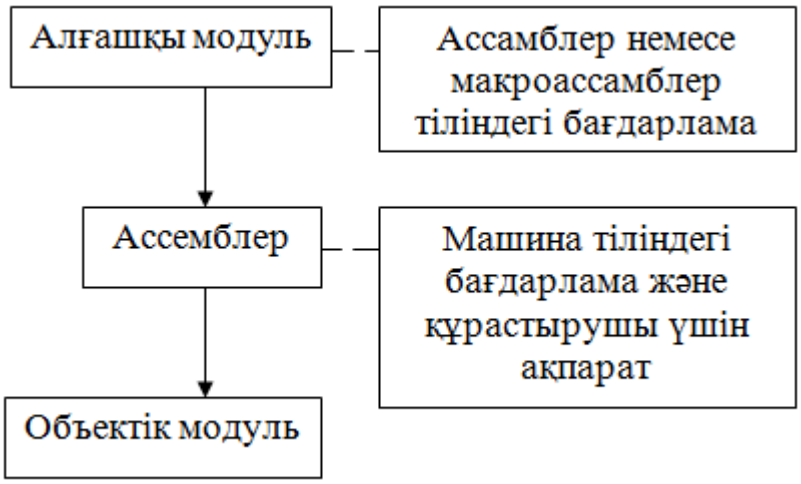


Сурет 8. Трансляциялаудың жалпы схемасы

Сонымен бастапқы модулді машина тілдегі немесе төменгі деңгейлі тілдегі объектік бағдарламаға түрлендіретін бағдарламаны *трансляциялау* деп атайды.

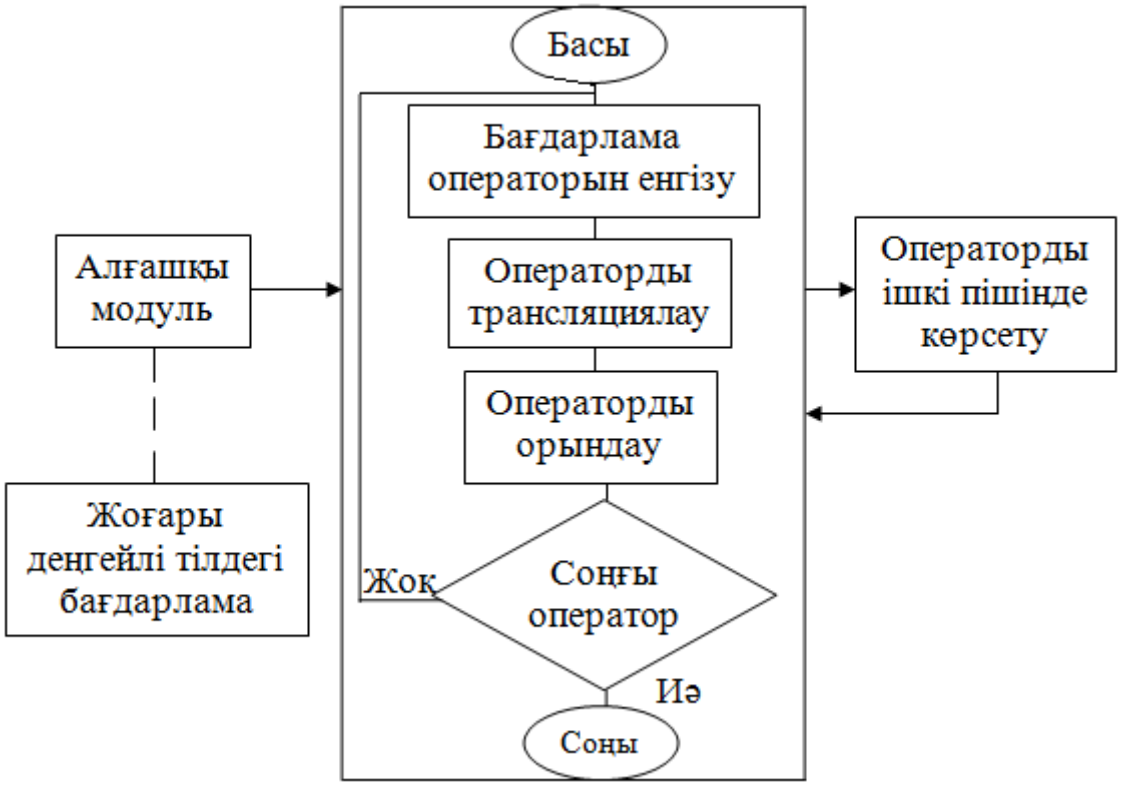
Ендірілетін тілге, трансляциялау кезеңдерінің өту ретіне және бағдарлама операторларының орындалуына байланысты транслятордың мынадай түрлері болады: ассемблер; компилятор; интерпретатор.

Ассемблер бастапқы модулді объектік бағдарламаның бір түрі болып табылатын объектік модулге түрлендіреді. Объектік модулдің оны оған тәуелсіз трансляцияланған басқа модулдермен біріктіруге және оның жедел жадтағы орналасуын реттеуге арналған машиналық командалары және ақпараттары бар. Объектік модул компьютерлерде тікелей орындалмайды, сондықтан да оны бағдарлама құрастырушының қосымша өңденуіне тура келеді. Ассемблер – бұл төменгі деңгейлі тілдің трансляторы, оның жұмысы төмендегі схемада бейнеленген (Сурет 9).



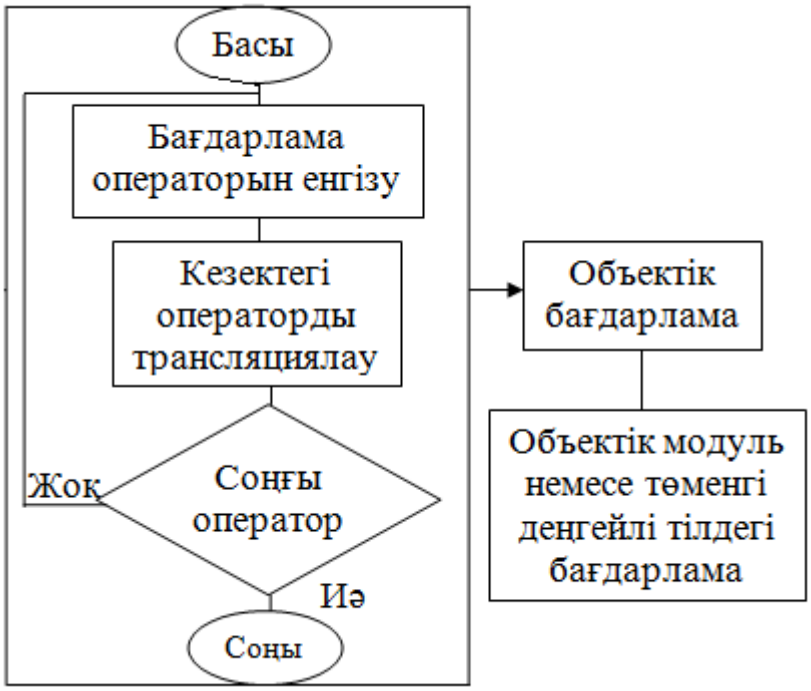
Сурет 9. Ассемблер

Жоғары деңгейлі тілдің трансляторы трансляциялау кезеңдерінің өту реттілігіне және бағдарлама операторының орындалуына байланысты ***интерпретатор*** немесе ***компилятор*** деп аталады. Интерпретатор әрбір жеке операторды трансляциялаған соң, оның тікелей тез орындалуын қамтамасыз етеді, яғни трансляция кезеңі мен орындалу кезеңі кезекпен қайталанады. Бағдарламаны интерпретациялау схемасы көрсетілген(Сурет 10).



Сурет 10. Интерпретатор

Компилятор бағдарламаның барлық операторларын трансляциялайды, ал бағдарламаның орындалуы жалпы алғанда оның қатынасуынсыз өтеді, яғни жеке операторларды трансляциялау кезеңдері тікелей бірінен соң бірі өтеді және бағдарламаның орындалуынан толық оқшауланған болады. Бағдарламаны компиляциялаудың қысқартылған схемасы (Сурет 11) көрсетілген.



Сурет 11. Компилятор

Жоғарғы деңгейлі тілдердің көпшілігі үшін компиляторлар жасалған. Интерпретаторлар жасалған тілдердің мысалына BASIC және FOCAL тілдерін атауға болады. Компьютерлердің бағдарламалау жүйесінде компилятордың екі түрі бар. Бірінші түрдегі компиляторлар бастапқы модульді машина тіліндегі объектік бағдарламаға түрлендіреді, яғни объектік модулді ассемблер тіліндегі объектік бағдарламаға түрлендіреді. Екінші түрдегі компилятор қалыптастырған осы бағдарламадан объектік модулді алу үшін қосымша *ассемблерді* пайдалану қажет болады.

2.3 Бағдарламалар кітапханасы

Бірнеше бағдарламада немесе бір бағдарламаның бірнеше жерінде белгілі бір әрекетке қол жеткізу үшін пайдаланылатын машина тіліндегі командалар тізбегін *бағыныңқы* бағдарлама деп атайды. Әртүрлі бағдарламаларда жиі қолданылатын бағыныңқы бағдарламаны бірыңғай ереже бойынша безендіріп, оны *стандартты бағыныңқы бағдарлама* деп атайды. Бағдарламаларды осылай стандарттау оларға бір пішінде ат беріп, онымен қарым-қатынас орнатуды, аргументтер мен нәтижелер туралы ақпаратты беру тәсілін формалды тіркеуді, сонымен бірге бағыныңқы бағдарламаны негізгі

бағдарламаға қосу мүмкіндігін автоматтандыруды қамтамасыз ететін бағыныңқы бағдарламалардың өздерін жасаудың бірыңғай ережелерін құруды көздейді.

Зауытта жұмысшы трактордың құрастырғанда оған қажетті болтты өз қолымен қалыпқа құйып, содан соң оған ойық бұранда шығарып жасап алу ойына да келмейді. Оның орнынына ол жанында тұрған сореден қолын созып керекті болтты таңдап алады. Егер басқа болт керек болса , онда ол оны басқа сөреден алады. Алдын ала даярланған стандартты бөлшектер жатқан мұндай сөрелер бұл кітапхана іспеттес.

Бағдарламалаушы адам да бағдарлама құрғанда осы жоғарыда айтылғанға ұқсас әрекет етеді. Үлкен бағдарламалардың 95 пайызы шағын ғана бұрын жасалып қойылған дайын стандартты бағыныңқы бағдарламалардан құрастырылады. Бүкіл дүние жүзі бойынша жазылған көптеген мың бағдарламалардың ішінен деректерді пернетақтадан ендіруді немесе ақпаратты экранға шығаруды басқаратын бағыныңқы стандартты бағдарламаны пайдаланбайтын жағдай өте сирек кездеседі. Бір кодты бірнеще рет қайта-қайта жазудың керегі не? Ішінен стандартты блоктарды алып, ешқандай өзгерістер ендірмей-ақ пайдалануға болатын бағдарламалар кітапханасында файлдар бар.

Мұндай кітапханалар жылдан жылға сұранысқа ие, әрі ірілене түсуде. Сондықтан да әр жыл сайын бағдарламалаушылардың еңбек өнімділігі арта түсуде.

Он бес жылдан астам уақыт бұрын компьютерлік бағдарламаның орташа өлшемі 40-50 Кбайтқа жететін. Ал он жыл бұрын ол 100 Кбайтқа жетті. Бағдарламалаушылар қазіргі кезде бірнеше Мбайт орын алатын бағдарламалар жасауда. Егер олар бағдарламалар кітапханасын пайдаланбайтын болса, онда мұндай бағдарламаларды даярлауға жиырма жылдай уақыт жұмсалған болар еді.

Компьютердің жадында тұрақты сақталатын стандартты бағыныңқы бағдарламалар жиыны стандартты бағыныңқы бағдарламалар кітапханасын құрады. Мұндай кітапхана құрамында бірнеше ондаған бағдарламалардан бірнеше жүздеген бағдарламаларға дейін болады. Әртүрлі есептерді шығаруда оларды қолдану мақсатында жиі кездесетін бағыныңқы бағдарламаларды жинау идеясы копьютерлердің өмірге келумен бірге пайда болды.

Жеткілікті толық бағыныңқы бағдарламалар кітапханасы бар жағдайда көптеген күрделі есептерді бағдарламалау есептелу процесін бірнеше кезеңдерге жіктеп, оларға бағыныңқы бағдарламалар кітапханасындағы дайын бағдарламаларды қолдануға және ол кезеңдерді үйлестіруге алып келеді.

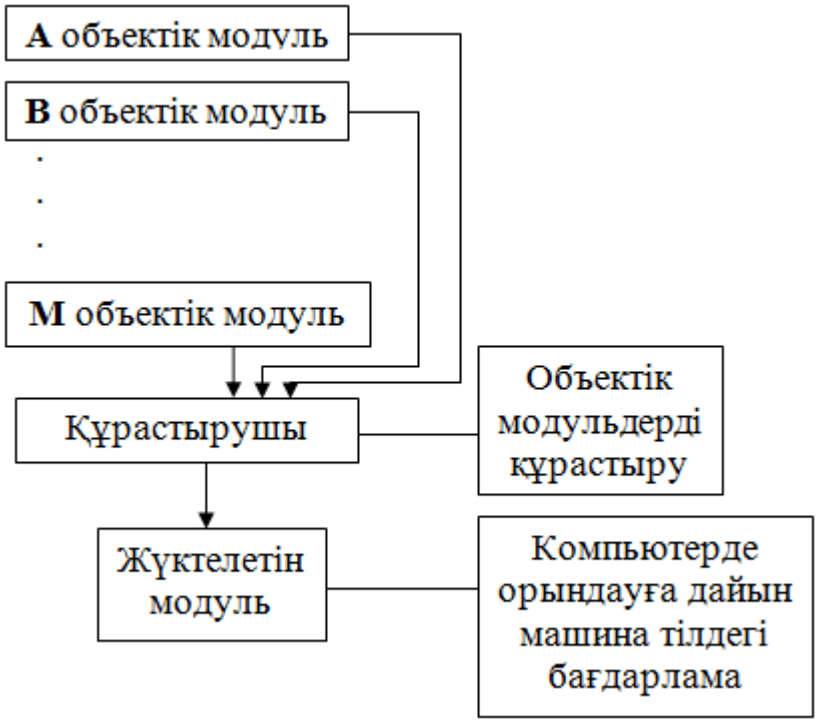
Сенімді жұмыс імтейтін *сыртқы есте сақтау құрылғыларын* жасау стандартты бағыныңқы бағдарламалар кітапханасын компьютердің жадында сақтауға мүмкіндік береді. Стандартты бағыныңқы бағдарламалар кітарханасы шын мәнінде компьютердің орындайтын операциялар жиынының бағдарламалық кеңейтілуі болып табылады. Осы операциялардың көпшілігін компьютердің аппараттық бөлігінің көмегімен жүзеге асыруға болар еді, бірақ бұл компьютердің құрылысын күрделендіріп жібереді.

Құрылып жатқан бағдарламаға компилятор математикалық функциялардың мәндерін есептейтін, енгізу-шығару операцияларын орындайтын объектік модулді қолдануды қосады. Бұл модулдер стандартты бағыныңқы бағдарламалар кітапханасында сақталады және одан *құрастырушы* бағдарлама автоматты түрде алады.

Қазіргі кезде стандартты, коммерциялық және фирмалық бағыныңқы бағдарламалар бар. Стандартты бағдарламалар кітапханалары кеңінен таралған. Әрбір бағдарламалаушы адам оларды басқа компаниялар арнайы коммерциялық прграммалар кітапханаларын жасаумен айналысып оларды басқа компаниялар мен жеке бағдарламалаушыларға сатумен айналасуда. Көптеген компанияларда жұмыс істеген жылдары жиналған өздерінің фирмалық бағдарламалар кітапханасы бар. Әрбір фирма безендірілуі, стилі және басқарылуы әр түрлі өздерінің жеке бағдарламалар кітапханасын пайдалануда. Фирмалық бағдарламалар кітапханалары таратылмайды және сатылмайды. Оларды бәсекелестерінен өте мұқият қорғайды. Бағдарламаны жасаушы фирмалар үшін бұл өте бағалы дүние болып есептеледі.

2.4 Құрастырушы және жөндеуші

Құрастырушы бірнеше объектік модульді операциялық жүйе жүктегеннен кейін компьютерде тікелей орындалу үшін мына төмендегі схемада(Сурет 12) көрсетілгендей даяр жүктелетін бір модуль етіп құрастырады.



Сурет 12. Құрастырушы

Мұндай құрастырудың қажеттілігін әрбір объектік модулде берілген модулге тәуелсіз трансляцияланған немесе стандартты бағыныңқы бағдарламалар кітапханасында сақталатын басқа объектік модулге өтудің барлығымен қамтамасыз етіледі.

Құрастырушы әрбір машина командасының және деректердің әрбір элементің операциялық жүйедегі өз орнын анықтайды және модульдердің бірімен-бірінің қатынас жасауын қамтамасыз етеді.

Пайдаланушы құрған бағдарламадағы қателіктерді іздеуді және жөндеуді тездетіп қысқарту үшін бағдарламалау жүйесінің құрамына *жөндеуші* деп аталатын бағдарлама ендірілген.

Әрбір *жөндеуші* бағдарлама нақтылы бағдарламалау тілінде жазылған бағдарламамен бірігіп жұмыс істеуге бағдарланған. Ол бағдарламадағы айнымалы мәндерінің өзгерістерін қарап шығуға, бағдарламаның орындалуын оперативті басқаруға, бағдарламаның орындалыушы операторын баспаға шығаруға, сонымен бірге басқа да қателіктерді іздеуді диалог режімінде жүзеге асыруға арналған бағдарламалық құралдарды береді.

Компьютердің оперативті жадына машина тіліндегі бағдарлама жүктеліп, компьютер оны орындауды бастағаннан кейін ғана компьютерде тікелей есеп шығару басталады. Машина тіліндегі бағдарламаның жұмысқа жарамды артық нұсқасы онда жіберілген қателіктерді тауып жөндейтін *жөндеушіден* өткеннен кейін ғана алынады.

Пайдаланушы модуліндегі қателіктерді екі түрге бөлуге болады:

- синтаксистік (пайдаланып отырған бағдарламалау тілінің синтаксисі бойынша жіберуге болмайтын модуль мәніндегі құрылымдар);
- мазмұндық (арифметикалық өрнектердегі дұрыс қойылмаған жақшалар, есептелу процесіндегі тармақталу шарттарының дұрыс тұжырымдалмауы, циклдің қайталану санының дұрыс берілмеуі т.с.с.).

Мазмұнындағы қателіктер шын мәнінде автор ойлаған алгоритмдік емес, бағдарламада басқа бір алгоритмнің орындалуына алып келеді.

Синтаксистік қателерді мәтінді синтаксистік талдау кезеңінде транслятор анықтайды және мәтіннен табылған қателіктің орнымен сипатын көрсететін диагностикалық хабар береді.

Мазмұнындағы қателіктерді анықтау үшін жоғарыда айтылғандай тест қолданылады.

Бағдарламаны жөндеу процесі кезінде бағдарламалаушыға аралық нәтижелерді беру қажет немесе кейбір командалардың орындалуының дұрыстығын тексеру керек. Бұл үшін бағдарламалау жүйесіндегі қызмет етуші жөндеушіні жұмысқа қосу керек. Бағдарламалаушы жөндеушіге тапсырма даярлау үшін арнайы тіл жасалады. *Жөндеуші* бұл тапсырмаларды компьютердің көмегімен орындайды. Ол жөнделіп жатқан бағдарламаның машиналық командалардың орындалуын қамтамасыз етуі және бақылау нүктелерінен өтуін қадағалауы тиіс.

Бақылау нүктесі – бұл мына төмендегідей қосымша әрекеттер жасау керектігін білдіретін бағдарламадағы нүкте:

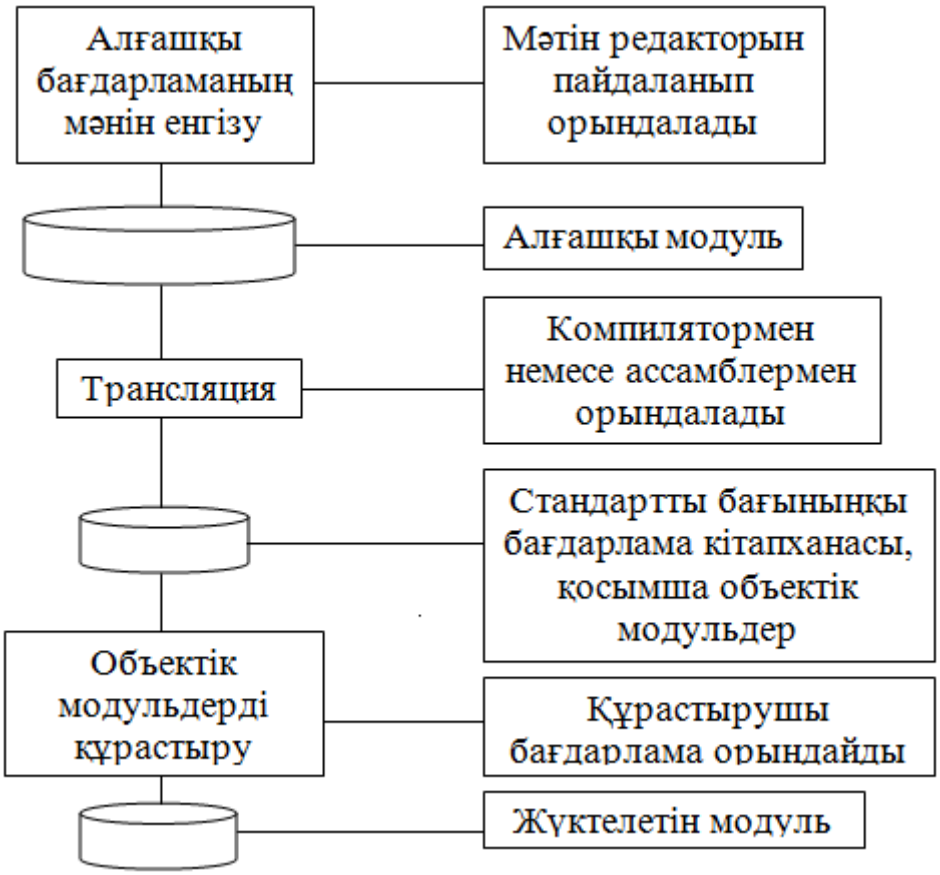
- аралық нәтижелерді баспаға шығару;
- бағдарламаның кезектегі бөлігіне қажетті бастапқы деректерді беру;
- бағдарламада қарастырған командалардың орындалу реттілігін өзгерту т.с.с.

Егер кезектегі орындаған (немесе орындалуға тиісті) команда бақылау нүктелерінің бірі болса, онда жөндеуші жөнделіп жатқан бағдарламаның орындалуын үзеді және осы бақылау нүктесі үшін берілген жөндеу әрекеттерін жүзеге асырады.

Жөндеуші екі бөлімнен тұрады: қайталаушы және өңдеуші.

Қайталаушы жөнделіп жатқан бағдарламаның командаларын кезекпеннен алып орындайды, бірақ олардың әрқайсысы орындалып болғаннан кейін, жөндеушінің екінші бөлігі өңдеуші бағдарламаға өтуді жүзеге асырады. Бұл бағдарлама таңдалынып алынған командада жөндеу үшін көрсетілген тапсырма бақылау нүктесінің бар ма немесе жоқ па? соны тағайындайды. Егер бақылау нүктесі болса, онда өңдеуші бағдарлама осы нүкте үшін берілген өңдеу әрекеттерінің орындалуын қамтамасыз етеді және бұдан соң қайталаушы бағдарламаға қайта оралу жүзеге асырылады. Жөндеуші жөнделетін тапсырма синтаксисінің дұрыстығын алдын ала тексереді. Синтаксистік қатесі бар тапсырмаларды жөндеуші орындамайды, оған сәйкес хабарларды баспаға шығарумен шектеледі.

Компьютерде бағдарламаны орындауға даярлаудың танымал схемасы (Сурет 13) төмендегідей болады. Бағдарламаны даярлау мәтін редакторын пайдаланып орындалатын бастапқы модуль файлын құрып қалыптастырудан басталады. Бұдан соң бастапқы модульді компилятордың немесе ассемблердің көмегімен трансляциялау орындалады. Трансляциялаудың нәтижесінде басқа объектік модульдермен бірге құрастырушы бір жүктелетін модулге біріктіретін объектік модул файлы пайда болады. Пайдаланушының командасы бойынша монитор жүктелетін модулді операциялау жүйесіне орналастырады және оның орындалуын қамтамасыз етеді.



Сурет 13 Модульді бағдарламалау

Басқа бағдарламалармен бірлесе отырып бірнеше рет жұмыс істеуге есептеліп жасалған, әрі тиісті түрде безендірілген бағдарлама *модул* деп аталады.

Стандартты бағыныңқы бағдарламаның өзі модул болып табылады, өйткені әрбір бағыныңқы бағдарламаны басқа бағдарламаны пайдалануға болады. Бағыныңқы бағдарламаның кемшілігі сонда, ол өзін шақырған бағдарламамен ғана жұмыс істейді, ал бағыныңқы бағдарламаны орындалуына қажетті барлық ақпарат сол шақырушы бағдарлама арқылы беріледі. Бұдан басқа кейбір жағдайларда бір бағдарламаның бірден бірнеше бағдарламамен бірге жұмыс істеу қажеттілігі пайда болады.

Модулдің бағыныңқы бағдарламадан айырмашылығы басқа модулдермен кеңінен жұмыс істеуге мүмкіндік береді. Бұл модул ұғымы бағыныңқы бағдарлама ұғымының одан әрі дамытылуы деген сөз, ал соңғысы модулдің дербес жағдайы болып табылады.

Қазіргі заманғы бағдарламалау жүйелері осы модулді бағдарламалауды ескеріп құрылады. Бағдарламалау жүйесінде модулдің үш түрі қолданылады:

- пайдаланушы модуль;
- жүктелуші модуль;
- абсолютті модуль.

Бұл үлгідегі модул адамға ыңғайлы бағдарламалау тілінде жасалады.

Пайдаланушы модулі екі бөлімнен тұрады: модул денесі; паспорт.

Модул денесі модулдің мәнін анықтайтын оның негізгі бөлігі болып табылады, ал паспорт - бұл оны түсіндіруші бөлігі. Паспортта модулді келесі кезекте қалай пайдалану туралы қажетті қосымша ақпарат болады.

Модулдер біреше қайтара пайдалануға арналғандықтан компьютерге арнайы ұйымдастырылған кітапханада сақталады одан қажет болуына қарай шақырылады.

Бағдарламаны алгоритмдік тілден машина тіліне аудару екі кезеңнен тұрады. Бірінші кезеңде модулді басқа модулдермен жұмыс істей алу күйіне келдіретін пайдаланушы модулінің пішінінен машиналық пішінге өту орындалады. Модулді ұсынудың осындай пішінді *жүктелетін модуль* деп аталады. Пайдаланушы модулінен жүктеуші модульге өту соған сай транслятордың көмегімен жүзеге асырылады. Әрбір модулді трансляциялау бар болғаны бір рет орындалады, одан соң ол кітапханада жүктелуші модул түрінде сақталады.

Екінші кезеңде жүктелетін модулді нақтылы бағдарламамен жұмыс істеуге икемдеу жұмысы орындалады. Бұл кезең жүктеу деп, ал орындалатын бағдарлама жүктеуші деп аталады. Компьютердің жадына модулді ендіру, оның жадтағы оған бөлінген орынға икемделіп орналасуын, сонымен бірге модулді берілген параметрлерге икемдеуді жүктеу деп түсінеміз.

Модулді жүктеу жаңа бағдарламаға модул қосылған сайын орындалатандықтан, жүктеу қарапайым әрі тез орындалу үшін жүктелуші модул машина тіліне мүмкіндігінше жақын болуы тиіс.

Жүктелуші модуль пайдаланушы модуль сияқты екі бөлімнен тұрады: модуль денесі; паспорт.

Модуль туралы қосымша ақпараты бар және оны жүктеуге пайдаланылатын паспорт жүктеушіге ыңғайлы пішімде ұсынылады.

Абсолютті модул, бұл жүктеу нәтижесінде алынған модул. Ол машина тілінде ұсынылады, жадта өз орнында және басқа модульдермен бірлесіп жұмыс істеуге икемделеді. Сондықтан да абсолютті модул компьютерде тікелей орындауға жарамды машина тіліндегі бағдарламаның бөлігі болып табылады.

Модулді бағдарламалаудағы нақты есепті шешуге арналған бағдарлама осы бағдарламаны құрайтын барлық модульдерді жіктеп, оларды біріктіру жолымен алынады. Егер бұл жағдайда кітапханада сақталуы дайын модулді пайдалануға болатын болса, онда тек жетіспендіктерін ғана қайта құруға тура келеді. Бұдан модулдердің бай кітапханасы бағдарламалауды жеделдетіп әрі қысқартатындығын көреміз.

Пайдаланушының компьютер көмегімен белгілі бір жұмыс атқаратын тапсырманы бағдарламалау жүйесінде тұжырымдап беру мүмкіндігі бар. Бұл үшін адамның жүйемен қарым-қатынас тілі пайдаланылады. Тапсырманың құрамында мыналар болуы мүмкін:

- трансляциялауға жататын пайдаланушы модулінің мәтіні;

қандай модулдерді трансляциялағанан соң кітапханаға жазу керектігі туралы ақпарат;

- жеке модулдерден, оның ішінде дайын модулдерден пайдаланушыны қызықтыратын бағдарламаны жинау туралы жүйеге берілетін нұсқау;
- алынған бағдарламаларды орындау туралы нұсқау.

Қазіргі заманғы, бағдарламалау жүйесі көп тілді болып табылады, яғни бағдарлама жазу үшін және оның әртүрлі модулдерін жазу үшін ең ыңғайлы әр түрлі бағдарламалау тілдері пайдаланылады.

Трансляторлар пайдаланушы модулін жүктелуші тілге аударады, сондықтан да бұдан әрі жүктелуші модуль қай модуль қандай транслятордың көмегімен алынғандығына қарамастан пайдаланыла беретін болады.

Жүктелуші модулдер компьютердің сыртқы жадындағы бір кітапханада сақталады. Әрбір модуль үшін кітапхана каталогында белгіленген модулдің аты, оның ұзындығы және кітапханадағы орны болады. Модулдің паспортын жеке сақтауға болады, сонда каталогта берілген модулдің паспортының ұзындығы және оның жадтағы орны туралы ақпарат болады.

Біріне - бірі сілтеме жасай толтырылған паспорттар болған жағдайда машиналық бағдарламаны алу үшін модулдерді жүктеу процесін негізгі екі кезеңге бөлуге болады. Бірінші кезеңде берілген бағдарламаны алу үшін жүктелуге жататын барлық модулдер айқындалады, бағдарламалық модулдер арасындағы жадты болу жұмысы атқарылады және әрбір модуль үшін барлық сыртқы және жалпы объектілердің шын адрестері анықталады.

Бұдан әрі модульдерді тікелей жүктеу жұмысы атқарылып, екінші кезең орындалады.

3 Паскаль бағдарламалау тілі

3.1 Тілдің негізгі элементтері

Бұл бағдарламалау тілін 70 жылдардың басында швейцария профессоры Никлаус Вирт ұсынған. Есептеу техникасының негізін қалаушы, ғалым Блез Паскальдың есімімен байланысты. Автор осы бағдарламалау тілі арқылы бағдарламалаудың негізін үйретуді мақсат еткен[1].

Паскаль тілінде бағдарлама дегеніміз есепті шешу алгоритмін орындайтын сөйлемдер тізбегі. Басқа тілдер сияқты бұл тілдің алфавиті, сөздігі, тіл құрылымына байланысты грамматикасы, синтаксисі бар. Қазіргі кезде Паскаль тілі кез келген күрделі есептерді шығара алатын, кең таралған стандартты оқу тіліне айналды.

Кез келген бағдарламалық тіл символдардан, сөздерден, сөз тізбектері және сөйлемдерден (операторлардан) тұрады. Сөздер символдардан құралады, ал операторлар сөз тіркестерінен тұрады. Қазіргі кезде Pascal тілі құрылымының қарапайымдылығы мен игеруге жеңілдігінің арқасында бағдарламалаушылардың арасында кеңінен қолданылады.

Паскаль тілінің шығу тарихына тоқталатын болсақ. Алдымен, алгоритмдік тіл - Алгол пайда болды(1950-60 жылдары). Ал, Паскаль тілі, осы Алгол-дың мұрагері(жалғасы) ретінде дамуда. Қазіргі уақытта, 1979 жылы қабылданған Паскаль тілінің стандарты басқа тілдерден қалыспайды[2].

Қазіргі кезде көп тараған қуатты бағдарламалау жүйелерінің бірі – Free Pascal және ABC Pascal болып есептеледі. Бұл бағдарламалау жүйелерінің негізі ретінде стандартты Object Pascal бағдарламалау тілі алынған, сонымен қатар жүйе құрамына қолданушыға бағдарламаны теруге, өңдеуге ыңғайлы болу үшін жаң-жақты қызмет көрсететін қабықша(орта, редактор) орнатылған.

Тілдің алфавиті бағдарламаның элементерін құрастыруға қолданылатын символдар жиынынан тұрады. Оған төмендегі әріптер, цифрлар және арнай белгілер(символдар) жатады:

- A B C B ... ... … X Y Z, a b c d … … … x y z - үлкен (кіші) латын әріптері;
- араб цифрлары - 0 1 2 3 4 5 6 7 8 9;
- арнайы символдар - ? ! . : ; `` ’ [] {} () + - * / = < > ‘ ~ # @ $ * & % ^

Тілдің кейбір символдарынан мағыналы сөздер, сөйлемдер құралады. Паскаль тілінің сөздігі ретінде төмендегі ұғымдар қолданылады*:*

- литерал(хабарлама);
- қызметші сөздер;
- түсініктеме;
- идентификатор(атау);
- деректер.

Литерал - ол апострофқа алынған литерлер(символдар) тізбегі, яғни оны хабарлама ретінде қолданылады.

Түсініктеме – бағдарламаның бөліктерін түсіндіру үшін қолданылады. Паскаль тіліндегі түсініктеме – оң және сол жағынан фигуралық жақшаға алынған символдар тізбегі.

Хабарламаны немесе түсініктемені жазу үшін ұлттық алфавиттің әріптерін қолдануға болады. Ұлттық алфавит әріптері клавиатураның жоғарғы жағындағы цифрлар қатарына орналасқан, олар арнайы драйверлер қосылғанда ғана жұмыс істейді. Егер текст операторлардың ішінде болса, онда ол хабарлама оны апострофқа, ал операторлардын тыс болса, онда ол түсініктеме оны фигуралық жақшаға алынып жазылады(Кесте 2).

Кесте 2. Хабарлама мен түсініктеменің жазылуы

Хабарлама	*Түсініктеме*
Writeln(‘symma=’)	{оператор бөлімі}
Writeln (‘үшбұрыштың ауданы =’)	(*цикл денесі*)
Writeln (‘введите значение’)	{квадрат теңдеудің коэффициенттерін енгізу}
Ескерту Хабарлама немесе түсініктеме бағдарламаның орындалуына әсер етпейді.	

Идентификатор - деректердің, функциялар мен процедуралардың, жалпы бағдарламадағы кез келген объектінің атын белгілеу үшін қолданылады.

Идентификатор ұзындығы 255 символға дейін болады, бірақ алғашқы 63 символы ғана қабылданыды. Паскаль тілінде екі түрлі идентификатор қолданылады: *стандартты* және *қолданушы*.

Қолданушының идентификаторы ретінде *қызметші сөздерді* және *стандартты идентификаторларды* қолдануға болмайды. Қолданушы идентификаторы: латын әріптерінен, цифрлардан және кейбір символдардан құралады. Символдардың араларына бос орын қойылмайды және тек қана әріптен басталуы керек(Кесте 3).

Кесте 3. Қолданушы идентификаторының жазылуы

Дұрысы	*Қате*
Prim_1; zadanie1;	1zadanie {бірінші символ әріптен басталуы керек}
Read (X,Y,Z);	Begin{қызметші сөз, оны Beg_in жазуға болады}
Summ:= X+Y+Z;	Пример {ұлттық әріптер қолданылмайды, primer}
Write(‘қосындысы=’, S)	Pr 1 2{символдар арасында бос орын бар}

Қызметші сөздер - тілдің құрылымында анықталған мағыналы, мән беретін сөздер. Олар белгілі міндетті (қызметті) атқаратын ағылшын сөздері(Кесте 4). Қызметші сөздерді басқа мақсаттарға пайдалануға болмайды. Стандарты идентификаторлар мен қызметші сөздерді бас әріппен де, кішкентай әріппен де жазуға болады. Мысалы: Begin, begin, BEGIN – сөздері бір ұғымды білдіреді.

Кесте 4. Қызметші сөздер мен стандартты идентификаторлар

Қызметші сөздер								
and array begin case	end file for function	not of or packed	then to type until	div do downto else	if label mod nil	program record repeat set	var while with	const goto procedure
Стандартты функциялар мен процедуралар								
abs eof get	eoln ord new	pred sqrt readln	succ trounc page	sin arctan put	chr exp *read*	ln round write	Odd Sqr reset	Cos Wrtiteln rewrite

Деректер - бұл ұғымның мағынасын тереңірек түсіну үшін мынандай сұрақтарға жауап беріп көрейік.

- Сұрақ: Компьютер немен жұмыс істейді?
- Жауап: Компьютер бағдарламамен жұмыс істейді.
- Сұрақ: Бағдарлама немен жұмыс істейді?
- Жауап: Бағдарлама деректермен жұмыс істейді.

Бағдарлама - әр түрлі типтегі деректерді өңдей алады, яғни сандарды, символдарды, суреттерді, дыбысты және т.б.. Паскаль бағдарламалау тілінде өндеу объектілеріне (деректерге) мыналар жатады(Сурет 14).

Сурет 14. Деректердің түрлері

Айнымалы - бағдарламаның орындалу барысында мәні өзгеріп отыратын шама. Тұрақты - бағдарламаның орындалу барысында мәнін өзгертпейтін шама. Деректердің қарапайым типтеріне сандар, символдық және логикалық шамалар жатады.

Сонымен қатар, компьютердің жадысында алатын орнына байланысты: *тұрақты және өзгермелі(*динамикалық) *өлшемді* құрылымды деректерде болады. Соңғы кездері бағдарламалаудың объектіге бейімделген бағытының дамуына байланысты, деректердің жаңа құрылымы - объект ұғымы енгізілген.

Кез – келген бағдарламаның мәні(нәтижесі) болады, егер оған деректер берілсе. Сонымен, есептерді бағдарламалаудан бұрын, алдымен мынандай шартты талаптарға назар аудару керек:

- *Бірінші сұрақ: нені өндеу керек(не берілген)?*
- *Екінші сұрақ: қалай өндеу керек(нені табу керек)?*

Паскаль тіліндегі қолданылатын деректер типтерін екі топқа бөліп қарастырамыз: қарапайым және күрделі типтер(Кесте 5).

Қарапайым деректер. Егер деректер жеке мәндер, мәндер жиыны немесе жиындардың жиынын қабылдайтын қарапайым шамалармен анықталса, онда олар *қарапайым* деректер деп аталады (бір атауға бір мән сәйкес меншіктеледі).

Кесте 5. Паскаль тіліндегі деректер типтері

Қарапайым типтер(скалярлық)		Күрделі типтер(құрылымдық)	
бүтін	Byte, integer, word, longint, shortint	Жолдық	String
логикалық	Boolean (мәндері – true немесе false)	массивтер	Array
символдық	Char (ASCII – кестесі)	жиындар	Set
санаулы		файлдық	File
нақты	Real, single, comp, double, extended	жазба	record
нұсқағыштық	^ pointer		

Паскаль тілінде көбіне ондық санау жүйесіндегі сандар өнделеді. Он алтылық санау жүйесіндегі сандарды(0, 1,2, 3, 4, 5, 6, 7, 8, 9, A,B,C,D,E,F) жазу үшін олардың алдына $ символы қойылады: $1FA.

Сандар бүтін және нақты болып екі түрде беріледі(Кесте 6). Нақты сандардың бүтін бөлігі мен бөлшегін нүкте арқылы бөліп жазады:

45.21 немесе -8.178

Кейде нақты сандарды экспоненциалды түрде жазуға болады. Экспоненциалды бөлім E символынан(мантисса) тұрады:

5.18E+02 = 518 = $5.18*10^2$ немесе 10E-03 = 0.01= $10*10^{-3}$

Сандық типті деректерге арифметкалық: - қосу(+), көбейту(*), бөлу(*), алу(-) және бүтін типті деректерге арналған - div(санның бүтін бөлігі), mod(санның қалдық бөлігін) амалдары қолданылады.

Мысалы, 7/4 =1.75 ; 7 div 4 = 1; 7 mod 4 = 3;

Сандық типті деректерге қолданылатын арифметкалық амалдардан, стандартты арифметкалық және түрлендіру функцияларынан *арифметикалық өрнектер* құрастырылады.

Pascal-да есептеу процесін жүргізу үшін көптеген стандартты функциялар қолданылады. Стандартты функцияларды бағдарламада жазу үшін алдымен функцияның атын, содан соң жай жақшалардың ішінде аргументін көрсету керек.

Бағдарламада функцияның аргументінің мәнін берген кезде міндетті түрде функцияның анықталу облысын ескеру қажет. Мысалы теріс санның квадрат түбірі болмайды, бөлшектің бөлімі 0-ге тең болмау керек, т.с.с. Стандартты функциялардың аргументі кез-келген өрнектен тұруы мүмкін.

Мысалы: SQRT(3*X*X+5/8); ABS(25-X); INT(SQR(X)).

Тригонометриялық функциялардың аргументтері радиандық шамамен беріледі және басқа функциялары түрлендіру формулалары арқылы өрнектеледі.

Кесте 6. Сандық типті деректер

Бүтін сандық типтер			
Тип	Ұзындығы	Санның таңбасы/ Цифрлар саны	Мәндерінің диапазоны
integer	2 байта	-/+	-32768 .. 32767
shortint	1 байт	-/+	-128 .. 127
byte	1 байт	+	0 .. 255
word	2 байта	+	0 .. 65535
longint	4 байта	-/+	-2147483648 .. 2147483647
Нақты сандық типтер			
real	6 байт	11-12	2.9e-39 .. 1.7e+38
single	4 байта	7-8	1.5e-45 .. 3.4e+38
double	8 байт	15-16	5.0e-324 .. 1.7e+308
extended	10 байт	19-20	3.4e-4932 .. 1.1e+4923
comp	8 байт	19-20	-9.22e18 .. 9.22e18

Random(x) функциясы – (0,x) аралығындағы кездейсоқ бүтін типті санды береді, егер аргументі көрсетілмесе, онда нақты типтегі (0..1) аралығындағы кездейсоқ сан беріледі. Бұл функцияны пайдаланбас бұрын кездейсоқ санның генераторын(randomize) қосу керек.

Pascal-дің өрнектерін құрғанда тілдің келесі ережелерін сақтау керек:
- Өрнектің барлық(бөлшек) бөлігі бір қатарға жазылуы тиіс.
- Өрнектің барлық жақшалары жай жақша(дөңгелек) болуы тиіс.
- Қатарынан екі арифметикалық амалды(мысалы, +/) жазуға болмайды.
- Есептеулер солдан оңға қарай амалдардың орындалу ретімен жүреді.

Мысалы, арифметикалық өрнектердің жазылуы:

Математика тілінде	*Паскаль тілінде*
$0,5(x+7)(x+2)(x-3)$	0.5*(x+7)*(x+2)*(x-3)
$\frac{x_1}{-x_2}$	x1/(-x2)
$\cos^2 x$	sqr(cos(x))
$e^{\sqrt{2\sin 3x}}$	exp(sqrt(2*sin(3*x)))
$A^{(2+B)}$	Exp(2+B*Ln(A));

Символдық - типті деректер

Символдық типтегі деректер апострофқа алынған символдар(char) және жолдық шамалар(string). Мысалы, символдық деректер: ‘a’, ‘xyz’, ‘F2c’ . Символдық типті деректермен: салыстыру, біріктіру және т.б. жолдарды өндеу амалдар орындалады. Бұған ASCII таблицасындағы барлық символдар жатады.

Логикалық - типті деректер

Математиканың ерекше бөлімі логиканы ашқан ағылшын математигі Д.Бульдің есіміне байланысты логикалық типті деректерді буллевтік(bollean) деп атайды. Паскаль тілінде логикалық деректерге: true(ақиқат) және false(жалған) логикалық тұрақтылар жатады. Логикалық тұрақтылардан және операциялардан - *логикалық өрнектер* құралады

Логикалық тұрақты шамаларды көбіне true - 1 , false - 0 деп белгілейді. Логикалық тұрақтылардың әртүрлі мәндерінде орындалатын логикалық операциялардың нәтижесі ақиқат таблицасында(кесте 7) берілген

Кесте 7. Ақиқат таблицасы

a	*b*	*A and b*	*A or b*	*A xor b*	*Not(a)*
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Логикалық өрнек – қарапайым және күрделі шарттардан(Кесте 8) құралады, оның нәтижесі: true, false логикалық тұрақтылар болады.

Кесте 8. Логикалық өрнектер

Қарапайым шарттар				
Белгі	Түсініктеме	Мысалы		Нәтижесі
<	Кіші	a=4	a < 6	True
<=	Кіші немесе тең	b=25	b < =23	false
>	Үлкен	x=6	x >5	true
>=	Үлкен немесе тең	y=8.1	y >= 8	false
=	Тең	c=11	c = 10	false
<>	Тең емес	d=0	d <> 3	true
Логикалық операциялар(күрделі шарттар)				
AND	"ЖӘНЕ"	a=4	(a>2) and (a<6)	True
OR	"НЕМЕСЕ"	a=25 b =23	(a>10) or (b<10)	true
NOT	"ЕМЕС"	x=6	Not(x>5)	False
XOR	"НЕМЕСЕ "ЕМЕС"	y=1 b = 11	(y>10) xor (b<10)	false

Деректердің санаулы типін қолданушы анықтайды, оларды кейде шектеулі деп атайды. Санаулы типті анықтау үшін оның мәндері алдын ала белгілі болу керек. Мысалы, мәні ай аттарынан тұратын - Month санаулы типі:

```
Type
   Month=(january, february, march, april, may, june, ouly, august, seprember,
          october, november, decemder);
```

Немесе гүлдер тізімінен тұратын Flowers санаулы типінің жазылуы:

```
Type   Flowers=(rose, lily, iris, aster);
```

Паскалда интервалды типтерде бар. Мысалы, Num, Litter - интервалды типтердің сипатталуы:

```
Type
   Num=1..100;        {1-ден 100-ге дейінгі бүтін сандар}
   Litter='a'..'z';       {a-мен z -аралығындағы латын әріптері}
```

Деректер қалай ұйымдастырылу керек деген сұрақты алгоритмді өңдеуді бастамай тұрып шешіп алу керек. Алдымен, деректер типін таңдау қажет. Одан кейін осы деректерді ұсыну тәсілін анықтау керек. Деректерді ұсыну, компьютердің және бағдарламалық қамтамасының мүмкіндіктеріне байланысты болады.

Деректердің қарапайым түріне бүтін және нақты сандар, символдар, логикалық шамалар жатады. Қарапайым түрдегі шаманы анықтаудың негізгі белгісі былай: *атауы - мәні*.

Қарапайым деректер жиынтығы көмегімен алынған деректердің күрделі түрі - құрылымдық деп аталады. Деректердің құрылымдық түрі келесі негізгі белгілер бойынша жіктеледі: біртипті- біртипті емес, реттелген - реттелмеген, тіке қатынау - тізбекті қатынау, статикалық - динамикалық. Құрылымдық типтерге *массивтер, жолдар, жиында, жазбалар, файлдар* жатады.

Паскаль тілінде бағдарлама құрылымы: орындалатын әрекеттер тізбегі(операторлар) жазылған және осы әрекеттерді орындау барысында қолданылатын деректердің типтерін сипаттайтын екі бөлімнен тұрады.

Паскаль тіліндегі бағдарлама құрылымы:

```
Program аты;
  {сипаттама бөлімі}
begin
 {операторлар бөлімі}
 end.
```

Мұндағы, Program аты; - бағдарламаның тақырыбы. Бағдарламаның тақырыбы нүктелі үтір символымен аяқталады (аты - қолданушы идентификаторы). Сипаттама бөлімінде бағдарламада қолданылатын деректердің типтері анықталады(Кесте 9). Begin – бағдарламаның басы .

Операторлар бөлімінде бағдарлама орындайтын операторлар тізімі жазылады. End. – бағдарламаның соңы.

Паскаль тіліндегі бағдарламаның кеңейтілген құрылымы:

```
PROGRAM аты;
      uses          { модульдермен байланыс бөлімі;}
      label         { белгілерді сипаттау бөлімі;}
      const         { тұрақты шамаларды сипаттау бөлімі;}
      type          { типтер бөлімі;}
      var           { айнымалы шамаларды сипаттау бөлімі;}
     procedure      { ішкі бағдарламалар бөлімі;}
     function       { ішкі бағдарламалар бөлімі;}

BEGIN
      Оператор_1;
        Оператор_2;
         Оператор_3;
            ... ... ...
          Оператор_n-1;
        Оператор_n;
END.
```

Бір бағдарламада жоғарыда көрсетілген сипаттама бөлімдерінің бәрін бірдей қарастыру міндетті емес. Кейде қарапайым бағдарламаларда айнымалы бөлімі ғана болады. Бұл бөлімдер туралы қысқаша мәліметтер 1-ші кесте келтірілген. Кейбір бағдарламаларда бөлімдер бір-бірімен байланыстырып қолданылады. Мысалы, x - айнымалысының мәні num интервалды (1-ден a-ға дейінгі) типінде сипатталған, ал a – тұрақты шама(мәні – 100-ге тең)

```
Const   a=100;
Type    Num=1..a;
Var     X : num;
```

Паскаль тіліндегі бағдарлама құрылымына мысал, үш санның арифметикалық ортасын есептейтін бағдарлама.

```
program pr0;
  var
     a, b, c : integer;
           s: real;
begin
   writeln (‘ Деректерді енгіз ‘);
    readln(a, b, c);
     s:=(a+b+c)/3;
      writeln(‘ s=’ , s:8:4);
        readln
end.
```

Кесте 9. Сипаттама бөлімі

Сипаттама бөлімі	*Түсініктеме*	*Мысалы*
uses модулдер тізімі;	Егер бағдарламада басқа модулдердің (CRT, GRAPH т.б.) функциялары қолданылса, онда бағдарлама тақырыбынан кейін сол модульді көрсету керек.	Uses Crt, Graph, Printer;

label белгі_аты ;	**Goto** – көшу операторы арқылы белгіленген операторға көшу керек. Белгі ретінде *қолданушы идентификаторын,* сандарды қолдануға болады, олар label сөзінен кейін жазылады	label set, 2 ;
		begin
		set: Readln(x);
		Y:=10+5*X; Goto2;
		… … ...
		2 : writeln(y); goto set;
		End;

const аты=мәні;	тұрақты шама өзінің аты және мәнімен берілуі керек.	Const G=9.81; A=67; Otvet=’yes’; C=true;
type тип_аты=дерек_типі;	Стандартты және құрылымды типтерден өзгеше типтерді де анықтауға болады.	type massiv=array[1..10] of real;
Var атау : деректер типі;	айнымалының аты және оның қабылдайтын мәнінің типі анықталады. Егер айнымалылар бір типтес болса, онда біріктіріп жазуға болады	var d : integer; c,t : real; h : char; lg : boolean;
procedure function	Процедура және функция бөліміне ішкі бағдарлама жазылады. Ішкі бағдарлама деп өзінің аты бойынша шақырып алуға болатын бағдарламаның дербес бөлігін айтады.	

Бақылау сұрақтары

1 Бағдарлама дегеніміз не?
2 Бағдарламалау тілі ұғымын қалай түсінесіз?
3 Pascal тілінің алфавиті неден құралады?
4 Тұрақтылар және айнымалылар дегеніміз не?
5 Қызметші сөздер дегеніміз не?
6 Идентификатор дегеніміз не?
7 Хабарлама мен түсініктеменің айырмашылығы неде?
8 Деректер типі не үшін қажет?
9 Деректердің қандай типтері бар?
10 Деректердің стандартты типтері қалай сипатталады?
11 Қолданылатын амалдар неге тәуелді болады?
12 Бүтін типке қандай амалдар қолданылады?
13 Логикалық типке қандай амалдар қолданылады?
14 Символдық типке қандай амалдар қолданылады?
15Паскаль тілінде бағдарлама құрылымы қандай?
16 Бағдарлама құрылымындағы сипаттама бөлімі не үшін қажет?

3.2 Негізгі операторлар

Операторлар деп – бағдарламаның негізгі бөлігін құрайтын, компьютерге белгілі бір әрекетті орындату үшін қолданылатын бағдарламалық тілдің сөйлемін айтады. Операторлар бір-бірінен нүктелі үтір арқылы бөлініп жазылады. Pascal-дың операторлары құрылымына байланысты қарапайым және құрмалас болып екіге бөлінеді.

Қарапайым операторлар тек бір ғана оператордан тұрады. Оларға меншіктеу, енгізу, шығару, шартсыз көшу, процедураларды шақыру операторлары жатады.

Құрмалас операторлар белгілі бір ережемен топтастырылған бірнеше операторлардан тұрады. Операторлық жақшалардың ішінде бірнеше операторлардың біріктіріліп жазылуын құрмалас оператор дейміз. Бағдарламада құрмалас операторлар бір оператор тәрізді оқылады, бБірінің ішіне салынған құрмалас операторларда болуы мүмкін. Құрмалас операторлардың жазылу түрі:

```
Begin
 1-оператор;
  2-оператор;
   3-оператор;
          ... ;
    n-оператор
End;
```

Бос оператор деп – бағдарламаны орындау барысында ешқандай қызмет атқармайтын операторды айтамыз.

Меншіктеу операторы – берілген өрнектердің мәнін есептеп, оны айнымалыға меншіктеу үшін қолданылады. Ол кез келген бағдарламаның негізгі операторының бірі болып табылады.

Меншіктеу операторының жазылу форматы:

<идентификатор>:=<өрнек>;

Мұндағы, «**:=**» - символы `меншіктеу` деген мағынаны білдіреді. x:=3; {x айнымалысына 3-ті меншіктеу} Мысалы,

```
r:=5;
r:=r+1;
r:=8;
```

Идентификатордың және меншіктелетін өрнектің типтері сәйкес болу керек. Кейбір жағдайларда нақты айнымалыға бүтін мәнді меншіктеуге болады, ал бүтін айнымалыға нақты мәнді меншіктеу үшін, оны trunc функциялары арқылы түрлендіру қажет.

Есеп 1. Үш бүтін санның арифметикалық ортасын табатын бағдарлама

```
program pr0;
   var a, b, c : integer;
         s: real;
begin
```

```
a:=2;  b:=4;  c:=6;
s:=(a+b+c)/3
  end.
```

Осындай айнымалыларының мәндері берілген қарапайым бағдарламаларды, компьютерде орындауға болады. Бірақ та, s айнымалысының мәнін экранда көре алмаймыз, өйткені бағдарламада ``экранға шығару`` операциялары қарастырылмаған.

Деректерді енгізу - шығару операторлары

``Шығару``- операциясы компьютерден адамға информация тасуды қамтамасыз етеді. Деректерді экранға шығаратын write немесе writeln операторларының жазылу форматы:

write(өрнектер_тізімі) *немесе* writeln(өрнектер_тізімі)

Осы операторлардың(write-жазу) қайсы орындалса да, тізімдегі өрнектердің мәндері экранға шығады. *ln* – сөз тіркесі жаңа жол деген мағынаны білдіреді, яғни деректерді жаңа жолдан бастап шығарады. Мұндағы, өрнектер_тізімі - үтір арқылы тізілген айнымалылар, хабарламалар немесе арифметикалық өрнектер болуы мүмкін. Мысалы,

Бағдарлама құрылымы	Нәтижесі
f:=10;	
write(f)	10
write(f+2)	12
Write('бағдарламалау');	Бағдарламалау тілі
Write(' тілі')	
Writeln('бағдарламалау');	Бағдарламалау
Writeln('тілі')	тілі

Енді алдыңғы тақырыптағы, pr0 - бағдарламасының мазмұнын өзгертіп, деректерді экранға шығару жолдарын қарастырамыз:

а) s айнымалысының мәнін экранға шығару үшін;

```
  program pr0;
 var
   a,b, c : integer;
   s : real;
begin
 a:=2;  b:=4; c:=6;
 s:=(a+b+c)/3;
 writeln(s)
end.
```

ә) хабарлама арқылы өрнектің мәнін шығарған жағдайда;

```
writeln('ср. ариф.= ', s)
```

б) нақты айнымалының мәнін шығару форматын қолдансақ;

writeln(‘s=’, s:6:2)

Нақты айнымалының мәнін шығарудың белгілі форматы болады, мынадай форматта a:n:m беріледі. Мұндағы, a- айнымалының аты, n - жалпы саны(позициясы), m- үтірден кейінгісі

`Енгізу` - операциясы адамнан компьютерге информация тасуды қамтамасыз етеді. Мәліметтерді енгізу read немесе readln операторларының жазылу форматы:

read(айнымалылар_тізімі); *немесе* readln(айнымалылар_тізімі);

Осы операторды(read-оқу) қайсы орындалса да, компьютер бағдарлама жұмысын тоқтатып, айнымалылардың мәнін күтеді. Тізімдегі айнымалылардың мәндерін, қолданушы клавиатурадан береді. Енгізілген мәндер айнымалыларға меншіктеліп, бағдарлама өз жұмысын жалғастырады.

Жоғарыдағы, pr0 - бағдарламасының құрылымына клавиатурадан мәліметтерді енгізу жолдарын орналастырамыз:

```
program pr0;
var a, b, c : integer;
   s: real;
begin
  write(‘ введите первое число ‘);1
    readln(a);
  write(‘ введите второе число ‘);
    readln(b);
  write(‘ введите третье число ‘);
    readin(c);
      s:=(a+b+c)/3;
   write(‘ср. ариф. = ’ , s:8:4)
end.
```

Немесе бір операторды көмегімен мәліметтерді бір жолға енгізген жағдайда: readln(a, b, c);

3.2.1 Шартты және таңдау операторлары

Тармақталу алгоритмдерін бағдарламалау үшін if - операторы қолданылады. Шартты оператор: *if - егер, then - онда, else - әйтпесе* деген қызметші сөздерден құралады. Паскаль тіліндегі if - операторының толық және қысқа құрылымы 10-ші кестеде көрсетілген.

Кесте 10. If - операторы

құрылымы	Жазылуы	Оқылуы	Мысалы
қысқаша құрылымы	if <Л_Ө> then оператор_1;	*егер <Л_Ө> - тің мәні ақиқат болса, онда оператор_1 орындалады;*	егер a – ның мәні нольден үлкен болса, онда r:=r+a өрнегінің мәнін есептеу керек: if a >0 then r:=r+a;
толық құрылымы	if <Л_Ө> then оператор_1 else оператор_2;	*егер <Л_Ө> - тің мәні ақиқат болса, онда оператор_1 әйтпесе оперератор_2 орындалады;*	егер a - ның мәні жұп болса, оның кубын, әйтпесе оның квадратын есептеу керек: if a mod 2 =0 then k:= a*a*a else k:=sqr(a);
Мұндағы, Л_Ө - Логикалық өрнек, қарапайым немесе күрделі шарт болуы мүмкін.			

Кейде, есептің шығару жолы бірнеше варианттармен беріледі. Бұл жағдайды бірнеше шартты операторды пайдаланып жүзеге асыруға болады, яғни then және else қызметші сөздерінен кейін жаңа шартты оператор жазылады. Мысалы,

$$\begin{cases} -5, \text{егер } x < -5 \\ x, \text{егер } -5 \le x < 0 \\ 2x, \text{егер } 0 \le x < 3 \\ 6, \text{егер } x \ge 3 \end{cases}$$

Есептің шарты бойынша x-ке байланысты есептің төрт вариантты шешімін екіге бөліп жазамыз: x<0 және x>=0,
онда:

```
if x<0 then  if  x<-5   then y:=-5
                        else y:=x
       else   if  x<3  then y:=2*x
                        else y:=6;
```

Бірінің ішіне бірі жазылған шартты операторларды *құрмалас шартты оператор* деп атайды. Жоғарыда жазылған шартты операторларда then және else қызметші сөздерінен кейін бір-бірден шартты оператор қолданылған. Егер then немесе else сөздерінен кейін бірден артық оператор болса онда олар операторлар жақшасына алынады, оны құрмалас операторлар деп атайды.

Құрмалас операторларды немесе құрмалас шартты операторларды пайдаланып, If операторын кеңейтіп жазуға болады, бірақ, ELSE қызметші сөзінің алдында " ; " (нүктелі-үтір) белгісі қойылмайды.
Кеңейтіліп жазылған if операторының форматы:

```
IF <Л_Ө> THEN   begin  опер_1;  опер_2;  ...  опер_n  end
           ELSE begin  опер_1;  опер_2;  ...  опер_n  end;
```

Мысалы, a айнымалысының мәніне байланысты b, c айнымалыларының мәндерін анықтау керек.

```
If a<0 then begin b:=b+1; c:=c+1 end
          else begin b:=b-1; c:=c-1 end;
```

Есеп 2. Бір-біріне тең емес a,b,c бүтін сандары берілген. Сандарды өсу реті бойынша орналастыру керек.

```
program pr1; ( авс, асв, вас, вса, сав, сва түрінде ауыстыру керек)
  var a, b, c : integer;
begin
  writeln('значение a, b, c'); readln(a, b, c);
if b<c then if d<c then writeln(a,' ' ,b, ' ' ,c)
                   else if a<c then writeln(a,' ' ,c,' ',b)
                                else writeln(c,' ',a,' ',b)
        else if d<c then if c<a then writeln(b,' ',c,' ',a)
                                 else writeln(b,' ',a,' ',c)
                   else writeln(c,' ',b,' ',a)
writeln
end.
```

Кейде шарттың берілуіне байланысты бірнеше операторлар тізбегінің ішінен бір операторды орындау қажет болған жағдайда тандау операторын қолдануға болады(Кесте 11).

Кесте 11. Case - операторының құрылымы

Құрылымы	тандау элементі	Түсініктеме	Мысалы

Case <тандау индексі> of
<тандау элементі>
else
оператор
end;
	тұрақты_1 :
	оператор_1;
	тұрақты_2 :
	оператор_2;
	... ...
	тұрақты_n :
	оператор_n
		тұрақты шамалардан және операторлар тұрады
		Таңдау индексі нақты типтен басқа кез келген стандартты тип болуы мүмкін, тек индекстің типімен тұрақтының типі сәйкес болуы керек
			Апта күндерінің номеріне сәйкес аттарын шығаратын бағдарлама құрылымы
			1 - понедельник
			2 - вторник
			…
			7 - воскресенье.

Case <тандау индексі> of
тұрақты_1 : оператор_1;
тұрақты_2 : оператор_2;
тұрақты_3 : оператор_3;
... ...
тұрақты_n : оператор_n
else
оператор
end;
		Case - операторының орындалуы: егер таңдау индексінің мәні таңдау элементіндегі тұрақты шамалардың біреуіне тен болса, онда сол тұрақты шаманың мәніне байланысты оператор орындалады, содан кейін басқару тандау элементтерінен кейін орналасқан операторларға беріледі.	case n of
			1 : writeln('понедельник');
			2 : writeln('вторник');
			3 : writeln('среда');
			4 : writeln('четверг');
			5 : writeln('пятница');
			6 : writeln('суббота');
			7 : writeln('воскресенье')
			else
			writeln('нет такого дня');
			end;

Жоғарыдағы мысалдың толық бағдарлама құрылымы

```
Program case_prim;
  Var  n : integer;
 Begin
  Write(‘введите номер дни ‘); readln(n);
     case n of
       1 : writeln(‘понедельник’);
       2 : writeln(‘вторник’);
       3 : writeln(‘среда’);
       4 : writeln(‘четверг’);
       5 : writeln(‘пятница’);
       6 : writeln(‘суббота’);
       7 : writeln(‘воскресенье’)
    else   writeln(‘нет такого дня’); end;
end.
```

Бағдарламаның кез келген операторын алдына қос нүкте қойып белгілеуге болады, сонда бағдарлама жұмысын қойылған белгі бойынща goto операторы басқарады, бірақ белгілер қойылмас бұрын сипаттама бөлімінде анықталуы керек.

Мысалы, $\frac{1}{x}$ -тің мәнін есепте.

```
label 5;
var z, n, y: char;     x: integer;
 begin
  writeln(‘Вы будете работать с программой’);
  write(‘для ответа введите символы N или Y’);
 readln(z);
   if z=’N’ then goto 5;
      write(‘введите значение x=’);
       readln(x);
       if x<>0 then  writeln(‘результат =’,1/x)
                 else  writeln(‘деление на ноль!’);
      5:  end.
```

3.2.2 Циклдік операторлар

Бағдарламадағы қайталанбалы процесстерді «үшін», «дейін» және «әзірше» - деп аталатын алгоритмдік құрылымдарды пайдаланып ұйымдастыруға болады. Осы құрылымдарға сәйкес Паскальда үш түрлі: for(үшін), while(әзірше) және repeat(дейін) циклдік операторлар бар(Кесте 12). Қайталанбалы саны белгісіз, берілген шартқа(Л_Ө) байланысты орындалатын цикл құрылымдары: «Дейі*н*», «Әзірше» деп аталады.

Кесте 12. Циклдік операторлар

Операторлар	Жазылу форматы	Түсініктеме
For – операторы	For i:=a to b do оператор;	қадамы, +1-ге өзгереді.
	For i:= B downto A do оператор;	қадамы, –1-ге өзгереді.
Repeat – операторы	repeat оператор until <Л_Ө>;	операторлар логикалық өрнектің мәні ақиқат болғанға дейін орындалады
while – операторы	while <Л_Ө> do оператор;	Әзірше логикалық өрнектің мәні ақиқат болса, онда операторларды орында
Мұндағы: a - цикл басы; b немесе Л_Ө - цикл соңы; оператор - цикл денесі; i-циклдің қайталанбалы параметрі(цикл қадамы).		

Циклдегі қайталанылатын әрекет - цикл денесі: шартты, құрмалас немесе циклдік операторды өзі де болуы мүмкін. Қайталанатын әрекет(операторлар) саны бірден артық болса, онда оларды операторлар жақшасына алып жазады.

Циклдік құрылымдарға мысал, 1-ден 5-ке дейінгі сандардың квадраттарының қосындысын есептейтін бағдарлама фрагменті(Кесте 13).

Кесте 13. Циклдік құрылымдарға мысал

For – операторы	Repeat– операторы	while – операторы	Цикл қадамы	Цикл нәтижесі

S:=0;
For i:=1 to 5 do
S:=s+sqr(i);
	s:=0; i:=1;
	repeat
	s:=s+sqr(i);
	i:=i+1
	until i>n;
		s:=0; i:=1:
		while i<=n do
		begin
		s:=s+sqr(i);
		i:=i+1
		end;
			
			I:=1
			I:=2
			I:=3
			I:=4
			I:=5
				S:=0
				S:=0+1
				S:=1+4
				S:=5+9
				S:=14+16
				S:=30+25

For i:=5 downto 1 do S:=s+sqr(i);			 I:=5 I:=4 I:=3 I:=2 I:=1	S:=0 S:=0+25 S:=25+16 S:=31+9 S:=40+4 S:=44+1

Есеп 3. n санының көбейту таблицасын шығаратын бағдарлама құрылымы(for операторы қолданылған):

```
var n, i, t: integer;
  begin
    write('ввод число');   readln(n);
     for i:=1 to 10 do
       begin
         t:=n*j;
         writeln(n, '*', j, '= ',t)   end;
   end.
```

Кейбір есептердің шартына байланысты бірінің ішіне бірі салынған циклдердіқолдануға тура келеді, оны қос цикл деп атайды. Мысалы, 1-ден 3-ке дейін сандардың көбейту кестесін шығаратын бағдарлама құрылымын қарастырамыз(Кесе 14)

Кесте 14. Қос циклдік операторлары

	I:=1		I:=2		I:=3	

For i:=1 to 3 do
begin
For j:=1 to 10 do
Write(I ,' * ', j , ' = ', i*j);
Writeln
End;
	J=1
	J=2
	J=3
	J=4
	J=5
	J=6
	J=7
	J=8
	J=9
	J=10
		1* 1 = 1
		1* 2 = 2
		1* 3 = 3
		1* 4 = 1
		1* 5 = 2
		1 * 6 =3
		1* 7 = 1
		1* 8 = 2
		1 * 9 =3
		1*10=10
			J=1
			J=2
			J=3
			J=4
			J=5
			J=6
			J=7
			J=8
			J=9
			J=10
				2* 1 = 2
				2* 2 =4
				2* 3 = 6
				2* 4 = 8
				2* 5 = 10
				2 * 6 =12
				2* 7 = 14
				2* 8 = 16
				2 * 9 =18
				2*10=20
					J=1
					J=2
					J=3
					J=4
					J=5
					J=6
					J=7
					J=8
					J=9
					J=10
						3* 1 = 3
						3* 2 =6
						3* 3 = 9
						3* 4 = 12
						3* 5 = 15
						3 * 6 =18
						3* 7 = 21
						3* 8 = 24
						3 * 9 =27
						3*10 =30

Мұндағы і-айнамалысы сыртқы цикл параметрі, j – айнымалысы ішкі цикл параметрі.						

Ескерту

Қайталанбалы процессті ұйымдастырған кезде циклден шығу(циклдің соңы) жолы міндетті түрде болу керек. Бағдарлама орындалып жатқанда циклді тоқтату үшін [Ctrl+break] клавишаларын басуға болады.

Есеп 3. C - функциясының мәнін есепте

$$c=\begin{cases} d+a & ,d<1 \\ d & ,1\le d\le 3 \\ d-b & d\ge 3 \end{cases}, \quad \text{мұндағы} \quad a=\begin{cases} \sin(1+x), x\in(0,1) \\ n! \quad x\notin(0,1) \end{cases}$$

```
(n!=1*2*3*...*n)
  var    n, i, f, b : integer;
          a, x, d, c : real;
begin
  write('n='); readln(n);
  write('x='); readln(x);
  write('d='); readln(b);
  f:=1;
 for i:=1 to n do
       f:=f*i;
   if (0<=x) and (x<=1) then a:= sin(1+x)
                        else a:=f;
   if d<1 then c:=d+a
          else if d>3 then c:=d-b
                      else c:=d;
  writeln('c=', c:8;4)   end.
```

Есеп 4. Екі бүтін санның ең үлкен ортақ бөлгішін табатын бағдарлама.

```
 Var
   m, n, x, y : integer;
begin
  write(‘введите два натуральных числа’);  readln(m, n);
 x:=m;  y:=n;
while  (x<>y) do
 begin
  if  x<>y  then x:=x-y
            else y:=y-x;
end;
  writeln(‘нод чисел ‘,m,’ и ‘,n,’ равен ‘,x)
end
```

Бақылау сұрақтары

1 Операторлар жақша деген не және олар калай пайдаланады?
2 Бос операторлардың қызметі мен орындалуы.
3 READ операторы дегеніміз не және айнымалылары қалай енгізіледі?
4 LN жалғауы не үшін қолданылады?
5 WRITE операторымен нәтиже экранға қалай шығады?
6 WRITE операторымен сандық шамалар қандай форматтарда шығады?
7Тармақталу командалары дегеніміз не?
8 IF ... THEN ... ELSE операторларының қызметінің орындалуы.
9 GOTO операторының қолданылуы.
10 CASE операторларының қызметі мен жазылуы.

3.3 Ішкі бағдарламалар

Бағдарламаның әр бөлігінен шақырылып, бірнеше рет қайталап орындауға болатын, топталған операторлар тізбегін (бағдарлама бөлігін) ішкі бағдарлама деп атаймыз. Бағдарламалау тілі – Паскальда ішкі бағдарлама Procedure және function деп аталатын қызметші сөздер арқылы басталады. Процедура мен функциялардың құрылымы, негізгі бағдарламаға ұқсас.

Функция құрылымы:

```
Function атау(ф_п_т : типі) : типі;
     Сипаттама бөлімі;
begin
     операторлар бөлімі;
end;
```

Мұндағы, ф_п_т - типтері анықталған формальды параметрлер тізімі.

Мысалы, n! – санының факториалын есептейтін функция құрылымы.

```
Function fakt(n: integer): integer;
  Var
    I,f : integer;
```

```
begin
  f:=1;
   for i:=1 to n do
     f:=f*i;
     fakt:=f;
end;
```

Осындай ішкі бағдарламаны негізгі бағдарламадан немесе басқа да ішкі бағдарламалардан шақыру үшін *функцияға қатынау* командасын пайдаланады:

Функция_атауы(н_п_т);

Мұндағы, *н_п_т* - типтері анықталған нақты параметрлер тізімі. Нақты параметрлер типі, формальды параметрлер типімен сәйкес болу керек.

Есеп 5. $\frac{a!+в!}{c!}$ өрнегінің мәнін есептеу керек.

```
Var
 a,b,c : integer;  y : real;
Function fakt(n: integer): integer;
 Var
   I,f : integer;
 begin
    f:=1;  for i:=1 to n do  f:=f*i;
    fakt:=f;
 end;
begin
 readln(a,b,c);
  y:=(fakt(a)+fakt(b))/fakt(c);
 writeln(y:8:4)
end.
```

Функциямен жұмыс жасағанда мыналарды ескеру қажет:

- Функция бір ғана нәтиже береді (бірнеше мән берілсе де)
- Нәтиже функцияның атына меншіктеледі, функция бағдарламасында міндетті түрде меншіктеу операторы болу қажет, сол жағына функцияның аты меншіктелген.
- Функция тақырыбында функция нәтижесінің типі көрсетілуі керек.
- бағдарламалардан функцияны шақыру командасын өрнектің тікелей ішіне орналастыруға болады.

Процедура құрылымы:

```
procedure атау(ф_п_т : типі);
 сипаттама бөлімі;
begin
  операторлар бөлімі;
end;
```

Мұндағы, ф_п_т - типтері анықталған формальды параметрлер тізімі, яғни процедураның енгізу-шығару параметрлері.

Осындай ішкі бағдарламаны, негізгі бағдарламадан немесе басқа да ішкі бағдарламалардан шақыру үшін *процедураға қатынау* командасын пайдаланады: *Процедура_атауы(н_п_т);* Мұндағы, *н_п_т* - типтері анықталған нақты параметрлер тізімі, яғни процедураның енгізу-шығару параметрлері.

Мысалы, т натурал санының цифрларының қосындысын есептейтін бағдарлама бөлігі(т=2346, нәтижесі s=2+3+4+6=15)

```
Procedure sum(n: integer; var s : integer);   { параметрлі – мәнді процедура }
 Var
    p : integer;
 Begin
   S:=o;
   While (n<>0) do
      Begin
         P:=n mod 10;
         S:=s+p;
          N:= n div 10;
      End;
End;
```

Бұл процедураға қатынау үшін: Sum(m, z); Мұндағы, m-енгізу, z-шығару параметрлері.

Процедура үш топқа бөлінеді: *параметрсіз процедура, параметрлі процедура, параметрлі – мәнді процедура.*

Бір бағдарламада бірнеше ішкі бағдарламалар болу мүмкін, оларға негізгі бағадрламалардан немесе басқада да ішкі програмалардан қатынауға болады. Кейде, процедура немесе функция өзін-өзі шақыруы да мүмкін. Бұны *рекурсия* - дейді, ал ішкі бағдарламалар: рекурсивті процедура немесе рекурсивті функция деп аталады.

Мысалы, рекурсивті процедура құрылымына

```
 procedure recur;      { параметрсіз  процедура }
 begin
   writeln(' procedure');
    recur;
    writeln(' f'unction’)
end;
```

Бұл параметрсіз процедураға жатады, оған қатынау үшін процедураның аты ғана жазылады: *recur;* мұнда шығару операторлары ғана орындала береді, яғни процедурадан шығу жолы қарастырылмаған. Міндетті түрде ішкі бағдарламалардан шығу жолын қарастыру керек, әйтпесе бағдарлама жұмысы тоқтап қалады.

Мысалы, рекурсивті процедура құрылымына

```
procedure recur(n : integer);  { параметрлі  процедура }
 begin
   write(' pocedure');
    if n>1 then recur(n-1);
end;
```

Бұл параметрлі процедураға қатынау үшін, процедураның аты және жақшаға оның параметрінің мәні жазылады: recur(5);

Мысалы, Евклид алгоритмі бойынша екі санның ЕҮОБ(n,m) табуға арналған, рекурсивті функция құрылымы:

```
Function nod(n,m) : integer;
  Var
     D : integer;
Begin
   D:=n mod m;
   If d=0 then nod:=m
             else nod:=nod(m,d);
End;
```

Ішкі бағдарламаның сипаттама бөлімінде ішкі айнымалылар(локалды немесе жергілікті) сипатталады, олар тек ішкі бағдарламаға ғана тиісті болады. Ал, негізгі бағдарламада сипатталған айнымалылар бағдарламаның барлық бөлігіне қатыса алады, оларды сыртқы(глобальды немесе ауқымды) деп атаймыз.

Бағдарламалау өнерінде ішкі бағдарламалар маңызды роль атқарады. Бағдарлама көлемін азайтады және компьютер жадысын да аз орын алады. Жақсы ойластырылған ішкі бағдарламалардан жеке бағдарламалар кітапханасын дайындауға болады. Осындай мақсатпен есептеу орталықтарында процедуралар және функциялар кітапханалары құрылады, олар машина жадысында, дискілерде саяқталады.

Есеп 6. Екі айнымалының орнын ауыстыру процедурасы бар бағдарлама құрылымы.

```
 Program retteu;
   Var x,yz : integer;
  Procedure zamena(var a,b: integer);
    Var t:integer;
  Begin
     T:=a;
       a:=b;
         b:=t
  End;
```

```
Begin
   Read(x,y,z);
    If x>y then zamena(x,y);
      If y>z then zamena(y,z);
        If x>z then zamena(x,z);
    Write(x,' ',y,' ',z)
  End.
```

Бақылау сұрақтары

1 Қандай қызметші сөздер ішкі бағдарламаларды сипаттайды?
2 Pascal бағдарламалау тілінде Procedure қызметші сөзі нені білдіреді?
3 Pascal бағдарламалау тілінде Function қызметші сөзі нені білдіреді?
4 Pascal бағдарламалау тілінде Program қызметші сөзі нені білдіреді?
5 Pascal бағдарламалау тілінде Uses қызметші сөзі нені білдіреді?
6 Қандай айнымалылар бағдарламада кез келген бөлікте көрінеді?
7 Қандай айнымалылар блок ішінде ғана көрінеді?
8 Бір ішкі бағдарламадан басқа ішкі бағдарламаға қатынауға бола ма?

3.4 Деректердің құрылымды типтері

ЭЕМ-нің жадында сандардың бір-бірімен байланысқан көптеген мәндерін сақтау үшін индексті айнымалылар, яғни *массивтер* қолданылады. Математикадағы вектор, сызықты кесте және матрица массивтерге мысал бола алады. Массивтер бағдарламалауда бір және екі өлшемді массивтер ретінде кеңінен қолданылады.

Паскальда массивтің индексі(элементінің реттік номері) сандармен немесе әріптермен тік жақшаның ішінде жазылады. Мысалы: сызықты массивтер - a[2] , c[i], b[k]; екі өлшемді массивтің индекстері жақшаның ішінде үтірмен бөлініп жазылады. Мысалы: a[2,1], b[i,j], c[5,k].

Бағдарламада массив элементтерін енгізу үшін немесе оларды экранға шығару үшін циклдік операторлар қолданылады. Матрица элементтерімен жұмыс істеу үшін қос цикл пайдаланылады(Сурет 15/Қосымша В).

Мысалы, n -өлшемді a- массивінің элементтерін енгізу:

```
For i:=1 to  n do   Readln(a[i]);
```

Экранға элементтерді бір жолға бос орын арқылы шығару:

```
For i:=1 to  n do     Write(a[i],'  ');
```

Есеп 7. Бір өлшемді массивтің ең кіші элементтін табу алгоритміне жазылған бағдарлама құрылымы.

```
  Const    A : array[1..8] of real=(4.5,5.7,2.8,3.2,7.4,9.5,5.7,58);
  Var      I, m : integer;
Begin
   M:=a[1];
   For i:=2 to 8 do
```

```
    If a[i]<m then m:=a[i];
  Writeln('минимальный элемент=', m)  End.
```

Есеп 8. Бір өлшемді масивтің элементтерін кему реті бойынша реттеу

```
   Var
    a: Array [1..5] of Integer;
    i, j, M, N: Integer;
begin
    N:= 5;
  For i := 1 to N do    Read( a[i] );
   For   i := 1 to  N-1 do
   For   j := i+1  to  N do
      if  a[i] < a[j] then   begin M := a[i]; a[i] := a[j]; a[j] := M end;
    For i := 1 to N do
     Writeln( a[i] )
end.
```

Паскаль тілінде символдар тізбегінен тұратын *жолдық айнымалылар* кең қолданылады. Мысалы, адамдардың тегі, заттардың аттары, мәтіндер және т.б. Паскальда жолдарды өндеуге string типі қолданылады.

Жолдармен орындалатын операциялар: жолдарды біріктіру(+), жолдың ұзындығын анықтау, ойып алу, салыстыру(символдардың коды бойынша).

Жодық айнымалының сипаталуы:

```
Var
    айнымалы_аты : string[өлшемі];
```

Мұндағы, өлшемі – жолдың ұзындығы, ол 1-ден 255-ке дейін бола алады.

String типінің функциялары:

length(a) – a жолының ұзындығы ;

copy(a,n,m) – a жолынының n символынан бастап m символын ойып алу;

concat(a,b,c) – a және b жолдарын біріктіріп, c жолын алу,c:=a+b деп те жазуға болады

pos(a,b) - a символын b жолынан іздейді,егер болса оның позициялық номерін шығарады;

delete(a,n,m) – a жолының n символынан бастап m символын жояды;

insert(a,b,m) – a жолын b жолына m позициядан бастап орналастырады;

val(a,b) – a жолын b санына түрлендіреді;

str(b,a) – b санын a жолына түрлендіреді;

chr(n) – n кодының символын анықтайды;

ord(a) – a символының кодын анықтайды;

Мысалы, берілген жолдағы A әрпінің санын есептейтін бағдарлама фрагменті

```
 k := 0;
   For i := 1 to length(s) do
     If     s[i] = 'A'    then k := k + 1;
 Writeln( k )   end.
```

Есеп 9. Берілген жолды кері оқитын бағдарлама құрылымы

```
Var  s, s1: string;
        i: integer;
begin
 Writeln('введите строку');
   Readln( s );
  s1 := '';
  For i := 1  to length(s) do
  s1 := s[i] + s1;
 Writeln(s1)
end.
```

Есеп 10. Символдық массив берілген. Жолдардың ең үлкенін табатын бағдарлама құрылымы

```
Var
    s, s1, M : string;
           I : integer;
Begin
 Writeln('введите строку');
    Readln( s );
  s1 := '';  M := '';
  For i := 1 to length( s ) do
    If s[i] <> ' ' then s1 := s1 + s[i];
                    else   begin    if length(s1)>length(M)  then M := s1;  s1 := '   end;
   if length(s1)>length(M) then M := s1;
 Writeln( M )
end.
```

Паскальдағы жиындар

Математикада ортақ қасиеті бар, элементтер тобын жиын деп қарастырады. Мысалы, жазықтықтағы фигуралар жиынына -квадрат, ромб, дөңгелек, тікбұрыш т.б. жатады. Жиындармен орындалатын операциялар :

- жиындардың бірігу,
- жиындардың қиылысу,
- жиындардың айырымы.

Паскальда жиын ретінде шектелген бір типтегі әр түрлі реттелмеген элементтер тобын қолданады. Жалпы жиын элементтерінің саны 256-дан аспауы керек, егер жиында бірде-бір элемент болмаса, онда оны бос жиын деп атайды. Бағдарламалау кезінде жиынды қолданарда, ол сипаттама бөлімінде анықталады:

```
Var
   <жиын_атауы> :  set of  типі;
```

Мысалы, әртүрлі типтегі week, m1, m2, m3, god - жиындарының сипатталуы:

```
Type
   days=(mon, tue, wed, thu, fri, sun);
var
   week : set of days;
    m1,m2,m3 : set of char;
    god : set of 1980..1999;
```

Паскаль тілінде жиындарға қолданылатын операциялар:

«+» - жиындардың бірігуі:

«*» - жиындардың қиылысуы;

«-» - жиындардың айырымы;

«=,<>» - жиындардың тең немесе тең еместігін тексеру;

«<=,>=» - жиындардың тиістілігін тексеру (егер а жиынның элементтері b жиынында болса(тексеру шартының жазылуы a<=b), онда а жиыны b жиынына тиісті);

in – элементтің жиынға тиістілігін тексеру(x in a –бұл операция x элементі a жиынына тиісті екендігін тексереді.

Жиын элементтері жиын конструкторы арқылы беріледі немесе клавиатурадан енгізу үшін және элементтерді экранға шығарғанда көмекші айнымалылардың көмегін пайдаланамыз. Ол үшін келесі әрекеттер тізбегі қажет – жиынмен типтес айнымалыны аламыз да тексереміз: айнымалы жиында бар ма, егер бар болса онда ол осы жиынның элементі болады.

Жиынның конструктор арқылы берілуі - тік жақшаға алынып үтір арқылы тізілген элементтер.

Мысалы, [3,5,8,4,7]
[‘a’,’f’,’7’,’$’,’=’]

Есеп 11. A және B символдар жиыны берілген. Екі жиынның қиылысуын анықтайтын бағдарлама құрастыр.

```
var
   a, b, c  :  set of char;
        x : char;
begin
   a:=[‘p’,’r’,’o’,’g’,’r’,’a’,’m’];
   b:=[‘m’,’n’,’o’,’j’,’e’,’s’,’t’,’v’,’a’];
   c:=a*b;
   writeln(‘элементы множества c:’);
  for   x:=’a’ to ‘c’ do
  if x in c then write(x,’ ‘);
 writeln
 end.
```

Есеп 12. Белілген сандардан 2-ге және 3-ке бөлінетін сандар жиынын анықтап, олардың тиістілігін тексеретін бағдарлама

```
   Var      m, m2, m3 : set of 1..100;
            i, x, n  : integer;
Begin
M:=[ ];  m2:=[ ];  m3:=[ ];
Write('введите колич. элементов множ M  : n=');
   readln(n);
  Writeln('введите элементы ');
for i:=1 to n do
 begin  read(x);
        m:=m+[x] end;
 For i:=1 to 100 do
  begin
   if  i in m  then
       begin   if  i mod 2=0 then      m2:=m2+[i];
               if  i mod 3=0 then   m3:=m3+[i]
       end
  end
    end.
```

Кейде, өнделетін ақпарат әртүрлі типтегі деректерден құралады. Мысалы, адам туралы мәліметтер: фамилиясы, аты, мекен жайы, жынысы, туған жылы. Мұндағы, алғашқы үшеуі символдық тип, төртіншісі санаулы тип, ал бесіншісі бүтін типтен болады. Мұндай жағдайда *жазба құрылымын* қолдану керек.

Жазба - деректер базазын ұйымдастыру кезінде кеңінен қолданылады, оның құрылымы кесте түрінде беріледі. Бұл деректер базасының реляциялық құрылымына жатады. Кестенің бағандарын - өрістер, ал жолдарын - жазбалар деп атайды. Мұндағы өрістер әртүрлі типтерден тұрады.

Паскальда жазбаның сипатталуы:

```
Type   тип _атауы =  record
                өріс_атауы1 : типі;
                өріс_атауы2 : типі:
                … … …
                end;
 var   жазба_атауы : тип _атауы
```

Мысалы;

```
Type person = record
        Fio, adres, sp : string;
        D_tel, r_tel : integer;
       End;
Var
       Viza : person;
```

Жазба элементтерімен жұмыс жасау үшін: *жазба_атауы.өріс_атауы* нүкте арқылы көрсетіледі. Мысалы, *viza.fio* – бұл viza атауымен берілген жазбадағы fio айнымалысы.

Есеп 13. Бір адамның визиткасындағы мәліметтер туралы бағдарлама құрылымы.

```
  Type person = record
        Fio, adres, sp : string;
        D_tel, r_tel : integer;
        End;
  Var
      Viza : person;
Begin
 Viza.fio:=’Никлау Вирт’;
 Viza.adres:=’г. Костанай’;
 Viza.sp:=’программист';
 Viza.d_tel:=233223;
 Viza.r_tel:=544545;
Writeln(‘фамилия- ’,Viza.fio);
Writeln(‘специальность-', Viza.sp);
Writeln(‘дом. Тел-', Viza.d_tel,' | раб. Тел-', Viza.r_tel);
Writeln(‘адрес-',Viza.adres);
Writeln
End.
```

Жазбаның атын қайта-қайта термеу үшін, біріктіру with операторын қолдануға болады. Жоғарыда берілген бағдарламадағы мәліметтерді экранға шығару жолдарын *with* операторының көмегімен түрлендірсек, онда бағдарлама фрагменті:

```
With viza do
  Writeln(‘фамилия- ’,fio);
     Writeln(‘специальность-', sp);
        Writeln(‘дом. Тел-', d_tel,' | раб. Тел-', r_tel);
          Writeln(‘адрес-',adres);
 End;
```

Бағдарламаны орындағанда қажетті деректерді клавиатурадан енгізіп, оларды экранға немесе принтерге шығаруға болады. Ал бағдарламаны қайтандан орындағанда сол деректерді қайтандан енгізуге керек. Бұл жағдай өте қолайсыз.

Сондықтан, кейде бағдарламаға қажетті деректерді және бағдарлама нәтижесін дискіде сақтаған жөн. Ол үшін файлмен жұмыс істеу керек. Паскальда файл және *файлдық айнымалы* деген ұғым бар. Файл – бір атаумен белгіленген әр түрлі типтегі деректер жиыны. Массивтен айырмашылығы: файл ұзындығы және элементінің номері анықталмайды.

Егер бағдарлама сыртқы файлдармен байланыс жасайтын болса, онда TYPE бөлімінде немесе VAR айнымлылар бөлімінде сипатталуы керек.

TYPE тип_атауы = FILE OF *Деректердің базалық типі* ;
VAR Файлдық айнымалы_атауы : тип_атауы;
Немесе
VAR Файлдық айнымалы_атауы : FILE OF *Деректердің базалық типі*;
Мысалы:

```
Type  mass=array[1..10]  of  integer;
      stroka=string[20];
 var   f1 : file of   mass;
      f2 : file of  stroka;
      f3 : file of  real;
```

Файлдық айнымалымен жұмыс істейтін функциялар:
- assign(f,‘a:\файл’)- f файлдық айнымалыны сыртқы файлмен байланыстырады.
- reset (f) - f файлы оқу үшін ашылады;
- rewrite(f) - f жазу үшін ашылады ;
- read(f,x) - ағымдағы нұсқағыш көрсетіп тұрған f файлының элементін x айнымалысына меншіктейді;
- write(f,x) – x айнымалысының мәнін f файлына жазады;
- close(f) - f файлын жабады.

Бір бағдарламада бірнеше файлдармен жұмыс істеуге болады, бір файлдағы мәліметтерді екінші файлға көшіруге немесе оларды жоюға болады. Файл текстік, типі анықталған және анықталмаған типтермен беріледі.

Қазіргі уақытта Паскаль бағдарламалау жүйесінде f - файлдық айнымалының типін сипаттамай, оның орнына базалық енгізу-шығару(input, output) жүйесін қолдануға болады.

Мысалы,

```
assign(input, ‘a.in’);
   reset (input);
assign(output, ‘a.out’);
  rewrite(output)
```

Есеп 14. f - файлына бүтін сандарды жазу керек. Төменде файлдық құрылыммен жұмыс істеудің екі түрлі нұсқасы берілген.

```
var  f : file of integer;
       i : integer;
begin
  assign(f, 'f.txt');
    rewrite(f);
    readln(i);
  while (i<>0) do
    begin
      write(f,i);    readln(i);
    end;
  close(f);   end.
```

```
var      i : integer;
begin
  assign(output, 'f.txt');
    rewrite(output);
    readln(i);
  while (i<>0) do
    begin
      write(i);  readln(i);
    end;
  close(output);
end.
```

Есеп 15 Теріс және он бүтін сандардан тұратын f файлы берілген. G –файлына он сандарды жазу керек.

```
var
   i : integer;
begin
  assign(input,'f.txt');
assign(output,'g.txt');
 reset(input);   rewrite(output);
  while  not(eof(output))  do
    begin
      read(i);
         if i>0 then    write(i);
     end;
  close(output);
 close(input);
  end.
```

Бақылау сұрақтары

1 Жазба құрылымын сипаттайтын қызметші сөз?
2 Жазба құрылымындағы біріктіру операторы?
3Файлдық құрылымды сипаттайтын қызметші сөз?
4Файлды оқу үшін ашу функциясы?
5 Индексі логикалық типті массив неше элементтен тұрады?
6 Қандай функцияның көмегімен берілген код бойынша символды анықтауға болады?
7 Length (A) функциясы не орындайды?
8 Pos(A,S) функциясы не орындайды?
9 Ord(‘A’) функциясы не орындайды?
10 Файлды жазу үшін ашу функциясы?
11 Copy(A, i, m) функциясы не орындайды?
12 VAL(A,B) процедурасы нені орындайды?
13 DELETE(A,N,M) процедурасы нені орындайды?
14 INSERT(A,B,N) процедурасы нені орындайды?
15 Жиын құрылымының сипаттайтын қызметші сөз?
16 Паскальда жиындарды біріктіру операциясы?
17 Элементтердің жиынға тиістілігін тексеру операциясы?
18 Массивтің базалық типі деген не ?

3.5 Тесттік тапсырмалар

1. $\cos^2 x^3$ өрнегінің дұрыс жазылған вариантын таңдаңыз?
A) cos(sqr(x)*x*sqr));
B) sqr(x)*cos(x*x*x);
C) cos(x*x*x)*sqr(x);
D) sqr(cos(x*sqr(x)))
E) sqr(cos(x)*sqr(x));

2. Берілген тексте A1 сөзін A2 сөзіне ауыстыру керек. (Сөздерді бөліктеу үшін бос орын пайдаланылады) Программаның дұрыс фрагмент таңда.

```
A)  var C,D:string;    I,K:integer;
 K:=1; D:='';
      For i:=1 to length(S) do
        If S[i]='' then
           Begin
            C:=copy(S,k,i-k);
             If C=A1 then begin D:=D+A2+S[i]; k:=i+1 end
                       Else begin D:=D+C+S[i]; k:=i+1 end
     End;
B) var C,D:string;  i,K:integer;
K:=1; D:='';
      For i:=1 to length(S) do
      If S[i]='' then
         Begin
           C:=copy(S,k,i-k);
            If C=A1 then D:=D+A2+S[i]
                       Else D:=D+C+S[i];
   End;
C) var C,D:string; I,K:integer;
K:=1; D:='';
      For i:=1 to length(S) do
        If S[i]='' then
           Begin
              C:=copy(S,k,k);
               If C=A1 then begin D:=D+A2+S[i]; k:=i+1 end
                         Else begin D:=D+C+S[i]; k:=i+1 end
End;
D) var C,D:string; I,K:integer;
K:=1; D:='';
     For i:=1 to length(S) do
      If S[i]='' then
         Begin
           C:=copy(S,k,i-k);
            If C=A1 then begin D:=D+A2+S[i]; k:=i+1 end
   End;
E) var C,D:string; i,K:integer;
K:=1; D:='';
      For i:=1 to length(S) do
        If S[i]='' then
            Begin
               C:=copy(S,k,i-k);
```

```
            If C=A1 then begin D:=A2+S[i]; k:=1 end
                    Else begin D:=C+S[i]; k:=i+1 end
    End;
```

3. Келесі конструкциялардың қайсы жиын болып саналады (Паскаль тілінің ережесі бойынша)
A) [[],[5]]
B) [‘*’,’*’]
C) [1.2 .. 3.1,5,6,7]
D) [2.5,2.8, .. 3.6]
E) [‘=’,’>=’,’>’]

4. Көрсеткіштік мәні болып табылады:
A) көрсеткіштер;
B) айнымалылар адрестері;
C) айнымалылар;
D) массивтер;
E) көрсеткіштердің адрестері.

5. Символдар түсін бояу процедурасы
A) TextBackground
B) TextColor
C) ReadKey
D) WhereX
E) Color

6. Input буферіндегі бір символды есептейтін функция
A) ReadKey
B) TextColor
C) WhereX
D) Function
E) TextBackground

7. Келесі программада не жазылады

```
int a[8]={1,2,3,4,5,-2,6,16};
main()
{  int i, p=1;
   for(i=0;i<8;i++)
    if(a[i] %2 ! =0) p*=a[i];
    printf(“%d”, p); }
```

A) 256
B)216
C)16
D)15
E)360

8. Процедура және функциялардан тұратын графикалық операциялардың орындалуын қамтамасыз ететін модуль
A) DOS
B) CRT
C) GRAPH
D) SYSTEM

E)SCR
9. Өрнектің мәнін есептеу керек
[3..14] * [4, 14..61] + [5..11] - [6..16] * [7]
A) [3.1..4.9, 5.5…7.5]
B) [4..6,8..11, 14]
C) [‘>=’,’<’,’=’]
D) [‘a’..’k’, ‘z’]
E) [5..9, 9.1..9.9]

10. (мұндағы 65014 және 65026 – массивтердің алғашқы адрестері a[2] және b[2]) массив көрсеткіштерін қолданған жағдайда келесі программа баспаға не басып шығарады
```
main()
{int i, a[2], *pa;
  long b[2], *pb;
  pa=a; pb=b;
  for(i=0; i<2; i++)
  printf(“указатели+%d: %u\n”, I, pa+i, pb+i);}
```
A) көрсеткіштер +1:65014 65026
   көрсеткіштер +2:65016 65028
B) көрсеткіштер +0:65014 65026
   көрсеткіштер +1:65015 65030
C) көрсеткіштер +0:65014 65026
   көрсеткіштер +1:65016 65030
D) көрсеткіштер +1:65014 65026
   көрсеткіштер +2:65015 65027
E) көрсеткіштер +0:65014 65026
   көрсеткіштер +1:65018 65030

11. Файлға 5 бүтін санды жазу үшін программаның дұрыс фрагментін таңда:
A)
```
rewrite(f);
for i:=1 to 5 do
begin
 read(n);
write(f,n);
end;
```
B)
```
rewrite(f);
for i:=1 to 6 do
begin
 read(n);
write(f,n);
end;
```
C)
```
rewrite(f);
for i:=1 to 5 do
begin
 write(f,n);
read(n);
end;
```
D)
```
rewrite(f);
for i:=1 to 5 do
begin
 read(n);
write(f,n);
```

```
end;
E) rewrite(f);
for i:=1 to 5 do
begin
 read(n);
write(f,);
end;
```

12. A[n] массивтің оң жұп элементтерінің санын анықтайтын программаның дұрыс фрагментін таңда

```
A) for(i=0;i<n;i++)
    If (a[i]>0 !! a[i]%2==0)
    k++;
B) for(i=0;i<n;i++)
    If (a[i]>0 & a[i]%2==0)
    k++;
C) for(i=0;i<n;i++)
    If (a[i]>0 && a[i]%2==0)
    k++;
D) for(i=0;i<n;i++)
    If (a[i]>0 && a[i]%2==0)
    k++;
E) for(i=0;i<n;i++)
    If (a[i]>0 && a[i]%2=0)
    k++;
```

13. Келтірілген программа баспаға қандай сандарды шығарады?

```
PROGRAM PAR;
VAR C,D:INTEGER;
PROCEDURE Q(X,Y:INTEGER);
PROCEDURE P(X,Y:INTEGER);
 BEGIN
   Y:=X+1 END;
BEGIN
C:=2; D:=0; P(SQR(C)+C,D);
       WRITELN(D);
C:=2; D:=0; Q(SQR(C)+C,D);
  WRITELN(D);
END.
```

A) 7 7
B) 2 3
C) 0 0
D) 6 7
E) 0

14. p^ типтелмеген көрсеткішке бейненің көшірмесі сақталынған. Программманың келесі фрагменті нені орындайды?

```
x:=10; y:=200;
while x<600 do
begin
  putImage (x,y,p^,xorput);
  delay(100);
```

```
  putImage (x,y,p^,xorput);
  inc(x,5);
end;
```
A) Суретті экранның сол жағынан оң жағына қарай жүргізеді.
B) Суретті экранның оң жағынан сол жағына қарай жүргізеді.
C) Нүктені экранның сол жағынан оң жағына қарай жүргізеді.
D) Суретті экранның төменгі шетінен жоғарғы шетіне дейін жүргізеді.
E) Сызықты экранның сол жағынан оң жағына қарай жүргізеді.

15. N бүтін сандардан тұратын тізімді құрастыратын программа фрагментін таңда. Типтелген көрсеткіш келесідей хабарланады:

```
Type
   Tp=^posl;
   Posl=record
   A:integer;
   Next:tp;
   End;
Var p,g,tp,i:integer;
```

A)
```
p:=nil;
for i:=1 to n do
begin
  new (g);
  read(g. a); g .next=p; p:=g;
end;
```

B)
```
p:=nil;
for i:=1 to n do
begin
  read(g^.a); g^. next=p; p:=g;
end;
```

C)
```
p:=nil;
for i:=1 to n do
  read(g^.a);
```

D)
```
p:=nil;
for i:=1 to n do
begin
new(g);
  read(g^.a); g^. next=p;
end;
```

E)
```
p:=nil;
for i:=1 to n do
begin
new(g);
  read(g^.a); g^. next=p; p:=g;
end;
```

16. n натурал сан берілген. a1,a2,…,a n бүтін сандар. Осы сандардың арасында беттесетін сандар бар екенін анықтау керек? Программаның дұрыс фрагментін таңдаю

A)
```
label M;
  Type t=array[1..n] of integer;
  Var
   a: t, i, j:integer;
  begin
```

```
  new(b);
 for i:=1 to n do
  read(a^[i]);
 for i:=1 to n-1 do
 for j:=i+1 to n do
  if (a^[j]=a^[i]) then begin
 writeln('ұқсас сандар бар'); goto M; end;
 writeln('ұқсас сандар жоқ');
 M:End.
```

B)
```
 label M;
 Type t=array[1..n] of integer;
 Var
  a: ^t, i, j:integer;
 begin
 for i:=1 to n do
  read(a^[i]);
 for i:=1 to n-1 do
 for j:=i+1 to n do
  if (a^[j]=a^[i]) then begin
 writeln('ұқсас сандар бар'); goto M; end;
 writeln('ұқсас сандар жоқ');
 M:End.
```

C)
```
Type t=array[1..n] of integer;
 Var
  a: ^t, i, j:integer;
 begin
  new(a);
 for i:=1 to n do
  read(a^[i]);
 for i:=1 to n-1 do
 for j:=i+1 to n do
  if (a^[j]=a^[i]) then begin
 writeln('ұқсас сандар бар'); goto M; end;
 writeln('ұқсас сандар жоқ');
 M:End.
```

D)
```
 label M;
 Type t=array[1..n] of integer;
 Var
  a: ^t, i, j:integer;
 begin
  new(a);
 for i:=1 to n do
  read(a^[i]);
 for i:=1 to n-1 do
 for j:=i+1 to n do
  if (a^[i]=a^[i]) then begin
 writeln('ұқсас сандар бар'); goto M; end;
 writeln('ұқсас сандар жоқ');
 M:End.
```

E)
```
Type t=array[1..n] of integer;
 Var
  a: ^t, i, j:integer;
```

```
begin
  new(a);
for i:=1 to n do
  read(a^[i]);
for i:=1 to n-1 do
for j:=i+1 to n do
  if (a^[i]=a^[i]) then begin
writeln('ұқсас сандар бар');
writeln('ұқсас сандар жоқ');
End.
```

17. Келесі есептер шешімдерінің дұрыс фрагментін таңдаңыз:
$x_1,....,x_8$ еспетеу керек $(x_1+x_2+2x_8)(x_2+x_3+2x_7)...(x_7+x_8+2x_2)$

A)
```
px=x; p=0
for(i=0;i<7;i+=2)
p+=*(px+i)+*(px+i+1)+2**(px+7-i);
```
B)
```
px=x; p=1
for(i=0;i<7;i++)
p*=*(px+i)+*(px+i)+2**(px+7-i);
```
C)
```
px=x; p=1
for(i=0;i<7;i++)
p*=*(px+i)+*(px+i+1)+2**(px+8-i);
```
D)
```
px=x; p=1
for(i=0;i<7;i++)
p*=*(px+i)+*(px+i+1)+2**(px+7-i);
```
E)
```
px=x; p=0
for(i=0;i<7;i+=2)
p+=*(px+i)+*(px+i)+2**(px+8-i);
```

18. Файлға 5 бүтін сандарды жазу үшін программаның дұрыс фрагментін таңда

A)
```
f=fopen("f.txt","w")
 for(i=1;i<=5;i++)
 {scanf("%d",&a)
  Fprintf(f,"%d",a);}
```
B)
```
f=fopen("f.txt","r")
   for(i=1;i<=5;i++)
    {fscanf(f,"%f",&a)
     printf("%d",a);}
```
C)
```
f=fopen("f.txt","r")
   for(i=1;i<=5;i++)
   {scanf("%d",a)
    fprintf(f,"%d",a);}
```
D)
```
f=fopen("f.txt","r")
   for(i=1;i<=5;i++)
    {scanf("%f",&a)
     fprintf(f,"%f",a);}
```
E)
```
f=fopen("f.txt","w")
   for(i=1;i<=5;i++)
   {scanf("%f",a)
    fprintf(f,"%f",a);}
```

19. A[17] массивінің жұп позицияларында орналасқан элементтерінің

арасындағы ең үлкен элементті анықтаудың дұрыс вариантын таңдаңыз.

A) MAX:=A[2];
FOR I:=2 TO 8 DO
IF A[2*I]>MAX THEN MAX:=A[I];

B) MAX:=A[1];
FOR I:=2 TO 17 DO
IF A[I]>MAX THEN MAX:=A[I];

C) MAX:=A[2];
FOR I:=2 TO 8 DO
IF A[2*I]>MAX THEN MAX:=A[2*I];

D) MAX:=A[2];
FOR I:=2 TO 17 DO
IF A[2*I]>MAX THEN MAX:=A[2*I];

E) MAX:=A[1];
FOR I:=2 TO 17 DO
IF ODD(I) THEN IF A[I]>MAX THEN MAX:=A[I];

20. Текстегі әріптерден басқа символдарды жою керек және барлық кіші әріптерді сәйкес үлкен әріптермен ауыстыру керек. Программаның дұрыс фрагментін таңда.

A)
```
for i:=1 to length (s) do
  If (ord (s[i])>=97) and (ord (s[i])<=122) then
  St:=St+upcase (s[i])
  Else
  If (ord (s[i])>=65) and (ord (s[i])<=90) then
  St:=St+s[i];
  Writeln (st);
```

B)
```
for i:=1 to length (s) do
  If (ord (s[i])>=97) and (ord (s[i])<=127) then
  St:=St+upcase (s[i])
  Else
  If (ord (s[i])>=65) or (ord (s[i])<=90) then
  St:=St+s[i];
  Writeln (st);
```

C)
```
for i:=1 to length (s) do
  If (ord (s[i])>=97) and (ord (s[i])<=127) then
  St:=St+upcase (s[i])
  Else
  If (ord (s[i])>=65) and (ord (s[i])<=90) then
  St:= s[i]+St;
  Writeln (st);
```

D)
```
for i:=1 to length (s) do
  If (ord (s[i])>=97) and (ord (s[i])<=127) then
  St:=upcase (s[i])
  Else
  If (ord (s[i])>=65) and (ord (s[i])<=90) then
  St:=s[i];
  Writeln (st);
```

E)
```
for i:=1 to length (s) do
  If (ord (s[i])>=97) or (ord (s[i])<=127) then
  St:=St+upcase (s[i])
  Else
  If (ord (s[i])>=65) and (ord (s[i])<=90) then
```

```
St:=St+s[i];
Writeln (st);
```

21. Көрсетілген программа баяндауларының Нәтижесін көрсетіңіз?

```
S:=0; P1:=1;
FOR I:=1 TO N DO
P1:=P1*I;
FOR I:=1 TO N DO
BEGIN
P2:=P2*K;
S:=S+1/(P2+1);
END;
S:=1/P1*S;
WRITE(S)
```

A) $\frac{1}{n!}\sum_{n!}^{1}\frac{1}{k!+1}$

B) $\sum_{n!}^{1}\frac{1}{(k+1)!}$

C) $\frac{1}{n!}+\frac{1}{k+1}$

D) $\frac{1}{n!}*\frac{1}{k!+}$

E) $\frac{1}{k!+1}$

22. a[10,10] массивтінің жұп бағаналарындағы теріс элементтеріне осы элементтердің квадраттарын меншіктеу керек.

A)
```
for j:=1 to 5 do
   for i:=1 to 10 do
      If a[i,2*j]<0 then a[I,2*j]:=sqr(a[I,2*j]);
```

B)
```
while j<=10 do
for i:=1 to 10 do
if  a[i,j]<0 then a[i,j]:=sqr(a[i,j]);
```

C)
```
j:=2;
while j<=10 do
for i:=1 to 10 do
if a[i,j]<0 then a[i,j]:=sqr(a[i,j]);
```

D)
```
for j:=1 to 5 do
 for i:=1 to 10 do
   if a[i,2*j]<0 then a[i,2*j]:=a[i,2*j]*2;
```

E)
```
for j:=2 to 10 do
for i:=1 to 10 do
if a[i,2*j]<0 then a[i,2*j]:=sqr(a[i,2*j]);
```

23. A[7.7] массивтерінің тақ жолдарда орналасқан оң элементтерінің қосындысы анықтайтын дұрыс вариантын таңдаңыз

```
A)For(i=0,s=0;i<7;i+=2)
        For(j=0;j<7;j++)
        If ([i][j]>0)s+=a[i][j];
B)For(i=0,s=0;i<7;i+=2)
        For(j=0;g<7;i++)
        If ([i][j]>0)s+=a[i][j];

C)For(i=0,s=0;i<7;i+2)
        For(j=0;j<7;j++)
        If (a[I,j]>0)s+=a[I, j];
D)For(i=0,s=0;i<7;i+=2)
        For(j=0;j<7;j++)
        If (a[i][j]>0)s+=a[i][j];
E)For(i=0,s=0;i<7;i+=2)
        For(j=0;j<7;j++)
        If (a[i][j]>0)s+=a[i][j];
```

24. F(m,n)=n!*m! M/(n+m)!, функцисын баяндау керек. n және m –щң сандар (факториалды есептейтін ішкі функцяны анықтау керек). Типтің баяндауы келтіріледі:

```
A) Function f(m,n:nat):real;
    Function fact (k:nat):nat;
Var I,p:integer;
Begin
 p:=0; for i:=2 to k do p:=p*I;
        Fact:=p
end;
Begin
 f :=fact(n)*fact (m)/fact(n+m)
 end;
B) Function f(m,n:nat):real;
    Function fact (k:nat):nat;
Var I,p:integer;
Begin p:=1; for i:=2 to k do p:=p*I;
        Fact:=p
Begin f :=fact(n)*fact (m)/fact(n+m) end;
C) Function f(m,n:nat):real;
    Function fact (k:nat):nat;
Var I,p:integer;
Begin p:=1; for i:=2 to k do;
 p:=p*I end;
        Fact:=p end;
Begin f :=fact(n)*fact (m)/fact(n+m) end;
D) function f(m,n:nat):real;
    function fact (k:nat):real;
Var I,p:integer;
Begin p:=1; for i:=2 to k do p:=p*I;
        Fact:=p end;
Begin f :=fact(n)*fact (m)/fact(n+m) end;
E) Function f(m,n:nat):real;
    Function fact (k:nat):nat;
```

```
Var I,p:real;
Begin p:=1; for i:=2 to k do p:=p*I;
        Fact:=p end;
Begin f :=fact(n)*fact (m)/fact(n+m) end;
```

25. A[5,5] (массив элементтерінің типі float) жолдық матрица шығаруларының дұрыс вариантын таңдаңыз

```
A) for(i=0;i<5;i++)
    for(j=0;j<5;j++)
    printf(“%f”,a[i][j]);
B) for(i=0;i<5;i++){
    for(j=0;j<5;j++)
    printf(“%f”,a[i][j]);
    printf(“\n”);}
C) for(i=0;i<5;i++){
    for(j=0;j<5;j++)
    printf(“%f”,a[i][j]);}
D) for(i=0;i<5;i++){
    for(j=0;j<5;j++)
    printf(“%f”,a[i][j]);
    printf(“\n”);}
E) for(i=0;i<5;i++){
    for(j=0;j<5;j++)
    printf(“%f”,a[i][j]);
    printf(“\n”);}
```

26. Келесі есептер шешімдерінің дұрыс фрагментін таңдаңыз:

a_1, ..., a_{10}; бүтін сандары берілген, теңдікті тексеру керек

$a_i+a_{11-i}>17$, i=1, …, 5

```
A)  px=*x
for (i=0;i<5;i++)
    If(*(px+i)+*(px+9-i)>17)
      {printf(“выполняется”); break; }
B)  px=x
for (i=0;i<10;i+=2)
  if(*(px+i)+*(px+9-i)<17)
    {printf(“не выполняется”); break; }
C)  px=x
for (i=0;i<5;i++)
     if(*(px+i)+*(px+9-i)<17)
       {printf(“не выполняется”); break; }
D)  px=*x
for (i=0;i<5;i++)
     if(*(px+i)+*(px+9-i)<17)
        {printf(“выполняется”); break; }
E)  px=x
for (i=0;i<5;i+=2)
      if(*(px+i)+*(px+10-i)<17)
        {printf(“не выполняется”); break; }
```

27.Келесі операторлардың дұрыс орындалуын көрсетіңдер:

```
S:=0; i:=1;
```

REPEAT
S:=s+1/(2*I|+1)
Dec(i);
UNTIL i<1;
A) 1/3
B) 1.0
C) 0.15
Д) 0.2
E) 0.75

28. Егерде бір өлшемді a массиві символьді айнымалылардан болса, онда қалай жазылады:
A) a:massive[1 . . 10] of char
B) a:array[1. . 10] of char
C) a:array[1. . 10] of real
Д) a;array[1. . 10] of char
E) a[1. . 10]:array of char

29. Программаны орындаудың нәтижесін көрсет:
Program pr;
Var
F, i:integer;
Function fb (n:integer):integer;
Else
Fb:=fb(n-1) + fb(n-2);
End.
Begin
N=5;
F:=fb(n);
Writeln(f)
End.
A) n
B)5
C)8
Д)3
E)0

30. 10 горизонтальды сызықты сызу қажет. Бағдарламаның дұрыс жауабын шығарыңдар.
A) Y:=10;
WHILE Y<=200 DO
BEGIN
LINE (50,Y,400,Y);
Y:=Y+20;
END;

B) Y:=10;
WHILE Y<=200 DO
BEGIN
LINE (50,Y,400,Y);
END;

C) Y:=10;
WHILE Y<=200 DO

```
BEGIN
LINE (50,Y,400,20);
Y:=Y+20;
END;
```

```
D) Y:=10;
WHILE Y<=200 DO
BEGIN
LINE (50,Y,400,Y);
Y:= 20;
END;
```

```
E) Y:=10;
WHILE Y<=200 DO
BEGIN
LINE (50,Y,400,2*Y);
Y:=Y+20;
END;
```

31.Бағдарламаны орындаудың нәтижесін көрсетіңдер:

```
x0:=0; x1:=1;
write (x0,’,’,x1);
FOR i:=1 to 5 do
BEGIN
        x:=x0+x1;
write(x,’,’);
x0:=x1; x1:=x;
end;
```

A) 0,1,1,2,3,4,5
B) 1,1,2,3,4,5
C) 0,1,1,1,1,1
D) 0,1,1,2,3,5,8
E) 1,2,3,5,8

32. Массивтің жұп элементтерінің санын және олардың қосындысын шығаратын программа фрагментін табыңыздар

```
A) s:=0; k:=0;
    for i:=1 to n  do
    if (a[i] mod 2= 0) then begin
    s:=s+a[i];
    k:=k+1;
    end;
```

```
B) s:=0; k:=0;
    for i:=1 to n  do
    if (a[i] mod 2= 0) then begin
    s:=s*a[i];
    k:=k+1;
    end;
```

```
C) s:=0;
    for i:=1 to n  do
```

```
    if (a[i] mod 2= 0) then begin
    s:=s*a[i];
    k:=k+1;
    end;
```

D)
```
    for i:=1 to n  do
    if (a[i] mod 2= 0) then begin
    s:=s+a[i];
    k:=k+1;
    end;
```

E)
```
    for i:=1 to n  do
    if (a[i] mod 2= 1) then begin
    s:=s+a[i];
    k:=k+1;
    end;
```

33 Сандық шаманы символдық шамаға ауыстыру операторы
A) STR
B) VAR
C) POS
D) DELETE
E) VAL

34 Файлдық тип ... қызметші сөзімен сипатталады:
A) ARRAY…OF
B) SET..OF
C) RECORD…END
D) FILE..OF
E) END

35. Программаны орындау нәтижесі
```
Programpr;
Var
a,b,c,d:integer;
u,p,s:integer;
function f(x,y:integer):integer;
begin
if(x>y) then f:=x
                else f:=y;
end;
begin
a:5; b:=-5; c:=2; d:=7;
u:=f(a,b);
 p:=f(u,c);
s:=f(p,d);
writeln(u,p,s)
end.
```
A) 5 2 7
B) u p s

C) -5 -5 -5
D) 5 2 7
E) 5 5 7

36 $\frac{e^x + a*b}{Cos^2 x + Sinx}$ Паскаль тіліндегі жазылуы
A) (EXP(1)+A*B) / (SQR(COS(X))+SIN(X)
B) EXP(X)+A*B) / (SQRT(COS(X))+SIN(X))
C) (EXP(1)+A*B) / (SQR(COS(X))+SIN(X))
D) (EXP(X)+A*B) / (COS(X)*COS(X)+SIN(X))
E) EXP^X+A*B) / (COS(X)*COS(X)+SIN(X))

37. Программаны орындау нәтижесін көрсет:

```
Var
   X,y:^integer;
Begin
   New(y); y^:=2*y^+sqr(y^);
    x:=y;
      write(x^+10);
Dispose(x);
End;
```

A) 15
B) x^+10
C) 35
D) 45
E) y+10

38. Алдын-ала шарт бойынша қайталайтын циклдік оператор келесі түйінді сөздерден тұрады:
A) WHILE…DO
B) REPEAT…UNTIL
C) WITH…DO
D) FOR…DO
E) IF…THEN

39. Курсорды келесі қатарға өткізу жұмысын орындайтын енгізу операторы
A) READ
B) WRITE
C) READLN
D) WRITELN
E) INPUT

40. Транслятор дегеніміз не
A) аудармашы
B) командалық қамтамасыз ету
C) барлық программаларды екілік жүйеге аударушы
D) машина кодына аудару жұмысын орындаушы программа
E) машина коды

3.6 Қарпайым бағдарламалау есептері

1	Үш натурал сан берілген. Олардың арасында тең сандар болмаса, «жоқ» деп, егер олардың екеуі бір-біріне тең болса, «иә» деп, ал үшеуі де бірдей сан болса, бәрі тең деп жазу керек.
2	a,b,c үш санның үлкенін табатын бағдарлама жаз.
3	Нақты a,в,c сандары берілген, a<в<c теңсіздігі орындалады ма? Тексер.
4	Екі үшбұрыш қабырғалары *a, b, c* және *d, e, f* берілген. Қайсы үшбұрыштың ауданы үлкен екенін анықтау қажет.
5	*a, b, c* және *d* төрт түзу сызығының ұзындықтары берілген. Осылар квадраттың немесе тікбұрышты төртбұрыштың қабырғалары бола алады ма? Соны тексеріңдер.
6	Үшбұрыштың қабырғалары *x, y* және *z* берілген. Осы үшбұрыштың тікбұрышты үшбұрыш екенін анақтау керек. Егер солай болып шықса, онда оның қай қабырғасы гепотенуза болатынын табыңдар.
7	P натурал сан берілген, a^p өрнегін есептейтін бағдарлама құрыңдар(бағдарламаны *while, repeat, for* операторларының әрқайсысын пайдаланып, шығаруға тырысыңдар)
8	Төмендегі өрнектердің мәнін табу қажет: а) $1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}, \quad n \geq 2$
9	ә) $\frac{1}{1*2}+\frac{1}{2*3}+\ldots+\frac{1}{(n-1)n}, \quad n \geq 2$
10	б) $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{(-1)n}{n}, \quad n \geq 2$
11	1-ден 100-ге дейінгі тақ сандардың және жұп сандардың қосындыларын табатын бағдарлама құрыңдар
12	Әрбір бактерия бір минутта екіге бөлінеді. Басында бір бактерия берілген деп, 5, 7, 9, ... 15 минуттан кейін неше бактерия пайда болатынын табыңдар. Жаңа пайда болған бактериялар да минут сайын көбейіп отырады.
13	12.. Z= $\begin{cases} \frac{c}{2+x}, & x>4 \\ 2c, & x \in [0;4] \\ x+a, & x<4 \end{cases}$ есепте мұндағы c= $\begin{cases} t, & t<2 \\ 2t, & t \geq 2 \end{cases}$, a= $\sum_{m=3}^{15} \frac{1}{k}$
14	$y_i=x_i+2.5$ функциясын есепте, $x_i=1+i*0.1$ былай анықталады(i=1,2, … 20).
15	n натурал саны берілген. Есепте $x_1=y_1=1$; $x_i=0.3x_{i-1}$; $y_i=x_{i-1}+y_{i-1}$; (i=2,3..)
16	Бір өлшемді массивтің ішіндегі индексі жұп болатын ең кіші элементті анықтау
17	Бір өлшемді массивтің ішіндегі индексі жұп болатын ең кіші элементті анықтау.
18	Бір өлшемді массивтің ішіндегі ең үлкен элементті анықтау

19	Бір өлшемді массивтің элементтерін өсу реті бойынша сұрыптау.
20	Бір өлшемді массивтің ішіндегі ең үлкен элементті алып тастау.
21	Екі өлшемді 5х4 массив берілген. Екінші жолдың бойынан ең кіші элементті анықтау.
22	Екі өлшемді 5х4 массив берілген. Төртінші бағанда орналасқан элементтердің көбейтіндісін анықтау
23	$a_1,\ldots,a_{10}$; $b_1,\ldots,b_{20}$ нақты сандары берілген. $[C_{ij}]$ i=1…20; j=1…10 нақты матрицасын алу керек. Мұндағы, $C_{ij} = \frac{a_j}{(1+\lvert b_i \rvert)}$
24	Екі өлшемді 3х3 массив берілген. Осы массивтің индекстерінің қосындысы тақ болатын элементтерді экранға шығару.
25	Екі өлшемді 4х5 массив берілген. Осы массивтің жұп элементтерін 0-ге, тақ элементтерін 1-ге айналдыру қажет
26	Матрицаның ең кіші элементі орналасқан баған элементтерін кему реті бойынша реттейтін бағдарлама
27	n натурал саны берілген. $\lfloor a_{ij} \rfloor$ (i,j=1,…,n) нақты матрицасын алу керек. Мұндағы, $a_{ij} = \begin{cases} \sin(i+j), & \text{егер } i < j \\ 1, & \text{егер } i = j \\ \sin\frac{i+j}{2i+3j}, & \text{қалған жағдайларда} \end{cases}$
28	Екі өлшемді 5х5 матрица берілген. Осы массивтің негізгі диагональ элементтерінің қосындысын табу
29	Екі өлшемді 4х4 матрица берілген. Осы массивтің кері диагоналінің астыңғы бөлігіндегі теріс элементтердің санын табу.
30	j*m және n*i өлшемді А және В матрицалары берілген. АВ матрицаларының көбейтіндісін табу керек
31	Екі өлшемді 5х5 матрица берілген. Осы массивтің негізгі диагоналінің үстіңгі бөлігіндегі нөлдердің санын анықтау
32	20 элементтен тұратын бүтін сандар массиві берілген. Осы массивтің әрбір төртінші орында тұрған элементтерін нөлге айналдыру
33	$[a_{ij}]$ i,j=1…n нақты квадрат матрицасы берілген. $[b_{ij}]$ i,j=1…n квадрат матрицасын алу керек. Мұндағы, $b_{ij} = \begin{cases} a_{ij}, \text{егер } j \geq i \\ a_{ji}, \text{егер } j > i \end{cases}$
34	20 элементтен тұратын бүтін сандар массиві берілген. Осы массивтің әрбір төртінші орында тұрған элементтерін нөлге айналдыру
35	4х5 матрица берілген. Матрицаның әрбір жолын өсу реті бойынша сұрыптаңыз.
36	a_1,a_2,a_3 бүтін сандар берілген. $[b_{ij}]$ i,j=1,2,3 бүтін санды матрицасын алу керек. Мұндағы $b_{ij} = a_i - 3a_j$
37	$x_1,x_2,\ldots,x_8$ нақты сандар берілген. 8-ші ретті квадрат матрицасын алу

	керек. $\begin{bmatrix} x_1 & x_2 & \dots & x_8 \\ x_1^2 & x_2^2 & \dots & x_8^2 \\ \dots & \dots & \dots & \dots \\ x_1^8 & x_2^8 & \dots & x_8^8 \end{bmatrix}$
38	Алмаста N алма бар. Сыныптағы M бала одан әрқайсысы 1 алмадан беруді сұрады. Алмасқа әр алманың салмағы қанша грамм екені белгілі. M баланың әрқайсысына бір алмадан бере отырып, Алмас өзінде қалған алмалардың (N-M) жалпы салмағы максимум болуын басты назарда ұстауы керек.
39	B(n,m) матрицасын алу керек. Матрицаның элементтері мына формуламен аңықталады: $b_{ij} = \dfrac{i + \sin\dfrac{\pi}{j}}{i - \cos\dfrac{(i+j)\pi}{i+2}}$
40	Төмендегі өрнекті есептеу: $Z=(a^n+b^m)c^r-d^k$
41	Берілген жолдың ішіндегі 3 символдан тұратын сөздерді экранға шығару.
42	n натурал саны берілген. $[a_{ij}]$ i,j=1,…,n нақты матрицасын алу керек. Мұндағы $a_{ij} = \cos(i^2 + n)$
43	Функциясының мәнін есепте: $\sum_{i=1}^{10}\sum_{k=1}^{n}\frac{\sin^k(x_i)}{k}$; мұндағы, x_i массив түрінде берілген ($x_1,x_2,\dots,x_{10}$) n=1,2,…,10
44	Берілген жолдың ішіндегі сөздердің санын анықтау
45	$\bar{a} = (a_0, a_1, ..., a_n)$, $\bar{b} = (b_0, b_1, ..., b_n)$ екі вектордың скалярлық көбейтіндісін есептеуге ішкі бағдарлама құру керек. Формуласы $C = \sum_{i=0}^{n} a_i b_i$ және бұл формуланы мына функцияның мәнін есептеуге қолдану керек: $z = \dfrac{\sum_{i=1}^{20} a_i b_i + 5\sum_{i=1}^{5} c_i k_i}{\sum_{i=1}^{20} a_i p_i}$

3.7 Олимпиада есептері

А есебі. Шеңбер

Енгізу файлының аты: A.in
Шығару файлының аты: A.out
Есептін жауабы файлының аты: A.{c,cpp,pas}
Уақыт шектеу: 2 секунд
Жадыға шектеу: 64 мегабайт

Шеңберде бірдей қашықтықта N нүктелер белгіленген, олар қарсы сағаттық тілдің бүтін сандармен 1-ден N-ға нөмірленген. Сіздерге осы шеңбердің бірнеше кос хордалар берілген, хорданың ұштары сол нүктелер болады. Әрбір қос хордалар үшін анықтаңыздар, олар қиылып кетеді ме, немесе киылымайды (сүйкелмейген хордалар киылмаған деп саналады).

Мәліметтерді енгізу форматы

Кіріс файлдың бірінші жолында екі бүтін сан тұрады: N және К ($1 <= N <= 10^9$, $1 <= К <= 100$). Келесі К жолда 4 бүтін сан тұрады: A_1, B_1, A_2, B_2 - бірінші хорданын (A_1, B_1) және екінші хорданың (A_2, B_2) нүктелерінің нөмірлері. Жолдағы сандар пробелмен арқылы бөлінген.

Мәліметтерді шығару форматы

Шығыс файлға әрбір қос хордалардың үшін бір жол жазылу керек - YES, егер хордалар киылып кетеді, немесе NO, егер хордалар киылыспайды (сүйкелмейді).

Мысал

A.in	A.out
4 3 1 3 2 4 1 2 3 4 1 2 3 2	YES NO YES

Есептің шешімі:

```
Program A_ecep;
label 1;
Var  A,B:array[1..4,1..4] of word;
    n:1..1000000000;
    k:1..100;
    i,j,max,min,max1,min1:integer;
 Begin
  assign(input,'A.in');reset(input);  assign(output,'A.out');rewrite(output);
     Readln(n,k);
   for i:=1 to k do
   for j:=1 to 2 do   read(a[i,j],b[i,j]);
  for i:=1 to k do
  for j:=1 to 2 do
```

```
 if ((a[i,j]>n) or (b[i,j]>n)) or ((a[i,j]=0) or (b[i,j]=0)) then goto 1;
  for i:=1 to k do
  begin  begin
   begin if a[i,1]>b[i,1]   then begin max:=a[i,1];min:=b[i,1];end
                        else begin max:=b[i,1];min:=a[i,1];end;end;
   begin if a[i,2]>b[i,2] then begin max1:=a[i,2];min1:=b[i,2];end
                        else begin max1:=b[i,2];min1:=a[i,2];end;end;
                        end;
 if max>max1 then begin
   if (min>=min1) and (min<=max1) then write('YES') else write('NO'); end;
   if max<max1 then begin
   if (min1>=min) and (min1<=max) then write('YES') else write('NO'); end;
  writeln;end;1:
  end.
```

B есебі. Бөлгіштер

Енгізу файлының аты: B.in
Шығару файлының аты: B.out
Есептін жауабы файлының аты: B.{c,cpp,pas}
Уақыт шектеу: 2 секунд
Жадыға шектеу**:** 64 мегабайт

А оң саны В санының бөлгіші аталып жатыр, егер В А санға қалдықсыз бөлінсе. Мысалы, 15 санында 4 бөлгіш бар: 1, 3, 5, 15. Әрбір берілген сандардың үшін, оның бөлгіштердің саны жұп сан немесе тақ сан болады ма сіздерге анықтау кажетті.

Мәліметтерді енгізу форматы

Кіріс файлдың бірінші жолында бір бүтін сан N ($1 < N <= 10^{s}$) жазылған. Келесі жолда N бүтін сан X_i ($1 <= X_i <= 10^{8}$) жазылған. Бір жолдағы сандар аралары пробелмен бөлінген.

Мәліметтерді шығару форматы

Шығыс файлда аралары пробелмен бөлінген N сан жазылу керек: і-ші сан 0 деп жазылады, егер X_i бөлгіші саны жұп сан болады, немесе 1, егер Хі бөлгіші саны тақ сан болады. ,

Мысал

B.in	B.out
2 4 5	1 0

Есептің шешімі:

```
program B;
Var x:array[1..10000] of longint;    a:array[1..10000] of word;
   n:1..100000;    m,i,j,r,k:integer;
Begin
  assign(input,'B.in');reset(input);   assign(output,'B.out');rewrite(output);
```

```
Readln(n);
for i:=1 to n do   read(x[i]);
for i:=1 to n do
 begin
      begin for m:=1 to x[i] do
      if x[i] mod m=0 then inc(k);end;
      inc(r);a[r]:=k;k:=0;
  end;
for r:=1 to n do
if a[r] mod 2=0 then write('0',' ') else write('1',' ');
end.
```

C есебі. Пайда

Енгізу файлының аты: C.in
Шығару файлының аты: C.out
Есептін жауабы файлының аты: C.{c,cpp,pas}
Уақыт шектеу: 2 секунд
Жадыға шектеу: 64 мегабайт

Компьютер процессор бөлігінен және монитордан құралады. Қоймада N процессор бөлігі және M монитор болып жатыр. i- ші процессор бөлігінің бағасы – A_i тугрик, j – ші монитордың бағасы – B_j тугрик . Дүниежүзілік қаражаттық дағдарыс артынын, i- ші процессор бөлігінен және j- ші монитордан құралған компьютердің бағасы $A_i \cdot B_j$ (көбейту) тугрик болады.Сіздерге дәл осылай компьютердің ең үлкен мүмкін саны жинау керек, олардың барынша ең көп мүмкін болған жиынтық бағасы болу үшін.

Мәліметтерді енгізу форматы

Кіріс файлдың бірінші жолында екі бүтін сан N және M (1 <= N,M <= 1000) . Екінші жолда N бүтін сан тұрады.i- жолдағы сан A_i . Үшінші жолда M бүтін сан тұрады: j- ші жолдағы сан B_j .(1 <= A_i ,B_j <= 1000) . Жолдағы сандар аралары пробелмен бөлінген.

Мәліметтерді шығару форматы

Шығыс файлға пробелмен бөлінген екі бүтін сан жазылу керек – ең үлкен мүмкін болған компьютердің саны және олрдың барынша ең көп мүмкін болған жиынтық бағасы.

Мысал

C.in	C.out
3 2 1 2 3 4 5	2 23

Есептің шешімі:

```
Program C;
Var a:array[1..1000] of integer;    b:array[1..1000] of integer;
   n,m:1..1000;   s:Longint;    i,j,r:integer;
```

```
Begin
 assign(input,'C.in');reset(input);
 assign(output,'C.out');rewrite(output);
  Readln(n,m);
  For i:=1 to n do    read(a[i]);
  for j:=1 to m do    read(b[j]);
  for i:=1 to n-1 do
  for j:=i+1 to n do
  if a[j]>a[i] then   begin r:=a[i]; a[i]:=a[j]; a[j]:=r;end;
  for i:=1 to m-1 do
  for j:=i+1 to m do
  if b[j]>b[i] then   begin r:=b[i]; b[i]:=b[j]; b[j]:=r;end;
  s:=0;
  if n>m then begin
           for i:=1 to m do  s:=s+(a[i]*b[i]);end
        else begin for i:=1 to n do  s:=s+(a[i]*b[i]);end;
  if m>n then write(n,' ',s) else write(m,' ',s);
 end.
```

D есебі . Жалқаулық

Енгізу файлының аты: D.in
Шығару файлының аты: D.out
Есептін жауабы файлының аты**:** D.{c,cpp,pas}
Уақыт шектеу: 2 секунд
Жадыға шектеу: 64 мегабайт

Оқытушы емтиханға дайындалуға үшін оқушыларына N сұрақтарды берді.Осы сұрақтардан ол емтиханға үшін А сұрақты таңдайды,ал оқушы, бесті алу үшін В сұрақты (бұларды А сұрақтан)жауап беру тиісті. Қу оқушы барлық сұрақтарды оқыуды қаламайды.Бесті алу үшін, сұрақтардың қандай ең аз саны оған жаттап алу керек ?

Мәліметтерді енгізу форматы

Кіріс файлдың жекеше жолында үш бүтін сан жазылған: N,A және B(1<=N<=100000, 1<=B<=A<=N).Сандар аралары пробел арқылы бөлінген.

Мәліметтерді шығару форматы

Шығыс файлда бір бүтін сан жазылу керек – есептің жауабы.

Мысал

D.in	D.out
10 7 3	6

Есептің шешімі:

```
Program D;
Var N:1..100000;    A,B,M:integer;  F,F2: text;
Begin
  Assign(f,'d.in'); reset(f);    Assign(f2,'d.out'); rewrite(f2);
```

```
  Readln(f,n,a,b);
  if (n>a) and (a>b) then  M:=n-a+b;
    writeln(f2,m);
  close(f);    Close(f2);
  end.
```

E есебі.Серіппе

Енгізу файлының аты: E.in
Шығару файлының аты: E.out
Есептін жауабы файлының аты: E.{c,cpp,pas}
Уақыт шектеу: 2 секунд
Жадыға шектеу: 64 мегабайт

N мөлшерілі серіппе – натуральды сандардылардың NxN мөлшерімен кестесі, кесте орталығында әрқашан 1 тұрады,одан оң жақтан 2, серіппе қпрсы сағаттық тілдің бұралып қалып жатыр. N мөлшерлі серіппесін шығарыңыздар.

Мәліметтерді енгізу форматы

Кіріс файлдың жекеше жолында бір бүтін сан жазылған – N (1<=N<100,N-тақ сан).

Мәліметтерді шығару форматы

Шығыс файлдың N жолда N бүтін саны жазу керек – N мөлшері спираль.Сандар аралары пробелмен бөлінген болу керек.

Мысал

E.in	E.out
1	1
3	5 4 3 6 1 2 7 8 9
5	17 16 15 14 13 18 5 4 3 12 19 6 1 2 11 20 7 8 9 10 21 22 23 24 25

Есептің шешімі:

```
Program E;
 Uses crt;
 label 1;
var i,j,m,n,l,r : integer;
    tab        : array [1..100,1..100] of word;
    begin
    assign(input,'E.in');reset(input);
    assign(output,'E.out');rewrite(output);
    readln(m);
    N:=m; l:=n*m+1;
```

```
  while l<>1 do begin inc(r);
  for i:=n-r+1 downto r do begin dec(l);tab[i,n-r+1]:=l end;
  if l=1 then goto 1;
  for i:=m-r   downto r+1 do begin dec(l);tab[r,i]:=l end;
  for i:=r   to n-r+1   do begin dec(l);tab[i,r]:=l  end;
  for i:=r+1 to m-r   do begin dec(l);tab[n-r+1,i]:=l end; end;
 1:
for j:=1 to m do
for i:=1 to n do
begin
 write(tab[i,j]:4);
 if i=n then writeln;
 end;
 end.
```

F есебі.Дәрежесі

Енгізу файлының аты: F.in
Шығару файлының аты: F.out
Есептін жауабы файлының аты: F.{c,cpp,pas}
Уақыт шектеу: 2 секунд
Жадыға шектеу: 64 мегабайт

Сіздерге A, B және C бүтін сандары берілген.A^B (A-ның B дәрежесі) C-ға бөлуінің қалдығы шығарыңыздар.

Мәліметтерді енгізу форматы

Кіріс файлдың жекеше жолында үш бүтін сан жазылған A ,B ,C ($0<=A,B<=10^{18}$,$1 <= C<= 10^{18}$). Санар аралары пробел арқылы бөлінген.

Есептің шешімі:

```
Program f;
Var a,b,c,e,d,i:Longint;
F1,F2: text;
Begin
  Assign(f1,'f.in');  reset(f1);     Assign(f2,'f.out'); rewrite(f2);
  Readln(f1,a,b,c);  d:=1;
  For i:=1 to b do
  If i>0 then  d:=d*a;
  E:=d mod c;
   writeln(f2,E);
  close(f1);    Close(f2);
  end.
```

Бөлім - II. Лабораторлық жұмыстарға нұсқау

Тақырып-1 Қарапайым сызықты бағдарламалау
Мақсаты: Сызықтық алгоритмдерді бағдарламалауды үйрету

Есеп - Төмендегі өрнектің мәнін есептейтін бағдарлама жазу керек:

$$\frac{2\sin\frac{x}{2}}{\sqrt{b}}$$

Паскаль тілінде

```
Uses crt;
{бағдарламада қолданылатын нақты типті айнымалылар сипатталған}
var x,y,b :real;
begin
 writeln('введите значение  X и  b'); {экранға хабарлама шығару}
  readln(x,b); {айнымалылардың мәндерін клавиатурадан енгізу}
   y:=(3*sin(x/2))/sqrt(b);  {өрнектің мәнін есептейді}
    writeln('y=',y:8:4); {экранға өрнектің мәнін шығарады}
readln   {бос оператор, enter клавишасын басуды күтуде}
end. {бағдарламаның соңы}
```

С тілінде

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
 {бағдарламада қолданылатын айнымалылар сипатталған, нақты тип}
float x,y,b ;
 {жаңа жолдан экранға хабарлама шығарады}
 printf("\n введите значение  X и  b\n");
 {айнымалылардың мәндерін клавиатурадан енгізу}
  scanf("%f%f",&x,&b);
   y=(3*sin(x/2))/sqrt(b);  {өрнектің мәнін есептейді}
    printf("y=%f",y); {экранға өрнектің мәнін шығарады}
getch();  {бос оператор, кез келген клавишаны басуды күтуде }
} {бағдарламаның соңы}
```

Тапсырмалар

1) $y = \frac{\sin(3x)}{\sqrt{2x^2+3}}$ 2) $y = \frac{\sqrt{2\cos(x)}}{x^4+0.001x}$ 3) $y = \frac{6x^3+2x}{\sqrt{6x+1}}$ 4) $y = \frac{\sin(2x)}{arctg(x^4)}$

5) $y = \frac{\sqrt{x^3+3x}}{2x+5}$ 6) $y = \frac{arctg(x)}{\sqrt{3x^2+5}}$ 7) $y = \frac{x^4+arctg(x)}{\cos(2x^3)}$ 8) $y = \frac{\sqrt{\sin(2x^4)}}{\cos(2x^3)}$

9) $y = \frac{\cos(4x-2)}{\sin(2x-4)}$ 10) $y = \frac{\ln(2x^4)}{\sqrt{3x^3}}$ 11) $y = \frac{tg(2x^3)}{3x^3+4}$ 12) $y = \frac{\sqrt{25x^3}}{\ln(x^2+3)}$

13) y = $\frac{tg(\sin(x))}{\ln(x^2+4)}$ 14) y = $\frac{\sqrt{3x^2+2}}{\ln(12x+3)}$ 15) y = $\frac{yg(2x+8)}{\ln(x3+5)}$ 16) y = $\frac{\sqrt{2x^4}}{\cos(2x^3)}$

Қосымша тапсырмалар

1 A(x_1,y_1) және B(x_2,y_2) координаталарымен берілген екі нүктенің арақашықтығын есептейтін бағдарлама жаз.

2 Мына өрнектің мәнін есепте: y = $\frac{tg(2x^3)}{3x^3+4}$

3 x,y,z берілген. u,v мәндерін табу керек.

U= $\frac{1+\sin(x+y)}{2+\frac{2x}{1+|\sin(x+y)|}}$ V= $\frac{x}{1+\sin(x+y)}$

4 Герон формуласымен үшбұрыштың ауданын есептейтін бағдарлама жаз.

5 x,y,z нақты сандары берілген, a,b есепте:

a= $\frac{3+e^{y-1}}{1+x^2\,|\,y-z\,|}$ b= 1+|y-x|+ $\frac{(y-x)^2}{2}+\frac{|\,y-x\,|^2}{3}$

Бақылау сұрақтары

1 Си және Паскаль тілдерінде бағдарлама құрылымы қандай?
2 Тіл алфавиті қандай топтарға бөлінеді?
3 Деректер типі дегеніміз не?
4 Енгізу функциясының қандай форматтары бар?
5 Шығару функциясының қандай форматтарын білесің?

Тақырып - 2 Шартты операторлармен жұмыс

Мақсаты: Тармақталған алгоритмдерді бағдарламалауды үйрету

Есеп - Төмендегі функцияның мәнін есептейтін бағдарлама жазу керек

Y = $\begin{cases} \sin(x^2) & , x <= 0 \\ x^3 & ,0 < x < 4 \\ 2x^2+3 & , x >= 4 \end{cases}$

Pascal – тілінде

```
Uses crt;
  var
  x,y :real;
 begin
 write('введите значение X= ');  readln(x);
{x айнымалысының мәніне сәйкес өрнекті есептейді}
 if  x<=0 then y:=sin(x*x)
           else   if x>=4 then y:=2*sqr(x)+3
```

```
                                    else y:=x*x*x;
 writeln('y=',y:8:4);
 readln
end.
```

C – тілінде

```
# include<stdio.h>
# include<math.h>
main()
{
 float   x,y ;
 printf("\n введите значение X = ");
 scanf("%f",&x);
    /*x айнымалысының мәніне сәйкес өрнекті есептейді */
 if ( x<=0)  y=sin(x*x);
     else  if (x>=4)  y=2*x*x+3;
              else y=x*x*x;
printf ("y=%f",y);
}
```

Тапсырмалар

1) $Y = \begin{cases} 2x^2+13 & ,x <= 0 \\ x^4 & ,0 < x < 4 \\ \sin(x^2) & ,x >= 4 \end{cases}$

2) $Y = \begin{cases} \sin(2x^2+2) & ,x <= 12 \\ x^3 & ,12 < x < 15 \\ \ln(2X) & ,x >= 15 \end{cases}$

3) $Y = \begin{cases} x^3+12x & ,x <= 2 \\ x^4 & ,2 < x < 3 \\ \sin(x^2) & ,x >= 3 \end{cases}$

4) $Y = \begin{cases} \sin(x^2) & ,x <= 0 \\ x^3 & ,0 < x < 4 \\ 2x^2+3 & ,x >= 4 \end{cases}$

5) $Y = \begin{cases} 2x^3+1 & ,x <= 2 \\ x^4 & ,2 < x < 3 \\ \cos(x^2) & ,x >= 3 \end{cases}$

6) $Y = \begin{cases} 3\cos(2x+2) & ,x <= 7 \\ x^2 & ,7 < x < 8 \\ \lg(x^3) & ,x >= 8 \end{cases}$

7) $Y = \begin{cases} \sin(2x+3) & ,x <= 4 \\ x^4 & ,4 < x < 5 \\ 12\ln(x) & ,x >= 5 \end{cases}$

8) $Y = \begin{cases} arctg(x) & ,x <= 6 \\ x^2 & ,6 < x < 8 \\ \lg(2x^2) & ,x >= 8 \end{cases}$

9) $Y = \begin{cases} \sin^2(x^2) & ,x <= 0 \\ tg(x^3) & ,0 < x < 1 \\ x^3 & ,x >= 1 \end{cases}$

10) $Y = \begin{cases} \cos(x^2) & ,x <= 12 \\ x^4 & ,12 < x < 13 \\ x^2+2x+2 & ,x >= 13 \end{cases}$

Бақылау сұрақтары

1 Тармақталу алгоритмдерінің құрылымдарының қандай түрлері бар?

2 Тармақталу операторлары қандай қызмет атқарады және олардың қандай түрлері бар?

3 Бос оператордың атқаратын қызметі қандай ?

Тақырып – 3 Циклдік операторлармен жұмыс
Мақсаты: Циклдік алгоритмдерді бағдарламалауды үйрету

Есеп - ҮШІН, ДЕЙІН және ӘЗІРШЕ циклдік құрылымдарын қолданып тізбектің алғашқы N мүшесінің қосындысын есептейтін бағдарлама жазыңыз.

$$\frac{\sin(n)}{2n+1}$$

Pascal – тілінде

```
 var
 n,i:integer; {i –цикл параметрі, n – тізбектің мүшелерінің саны}
   s:real    { s – тізбектің қосындысы}
begin
  write('n='); readln(n);
{ ӘЗІРШЕ – циклдік құрылымы, әзірше шарттың мәні ақиқат, ператорларды орында }
  s:=0; {басында қосынды нөлге тең болады}
  i:=1: {цикл басы}
  While  i<=n do { цикл соңы}
 begin
   s:=s+(sin(i)/(2*i+1)); {тізбектің  i-ші мүшесі }
   i:=i+1  {цикл қадамы, i-дің келесі мәні }
  end;
  writeln('While_Do -> S=',s:8:2);
{ДЕЙІН – циклдік құрылымы, шарттың  мәні ақиқат болғанға дейін операторларды орында }
  s:=0;  i=1;
  repeat
  s:=s+(sin(i)/(2*i+1));
  i:=i+1
  until i>n; {цикл соңы}
 writeln('Repeat_Until -> S=',s:8:2);
 {ҮШІН–циклдік құрылымы, +1(-1) қадаммен операторларды орындайды}
 s:=0;
{цикл басы, қадамы және соңы for – жолында анықталады}
 for i:=1 to n do
  s:=s+(sin(i)/(2*i+1));
  writeln('For_To_Do -> S=',s:8:2);
readln
end.
```

C – тілінде

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
main()
{ /*i –цикл параметрі, s –қосынды n – тізбектің мүшелерінің саны */
 Int   i,n;
 float  s;
   printf("\n n="); scanf("%d",&n);
```

```
/* ӘЗІРШЕ – циклдік құрылымы, әзірше шарттың мәні ақиқат,
операторларды орында */
  s=0;                /* басында қосынды нолге тең болады */
  i=1:               /*цикл басы */
While  (i<=n)        /*  цикл соңы*/
{
 S+=(sin(i)/(2*i+1));   /* тізбектің i-ші мүшесі */
   i++ ;                /*цикл қадамы, i-дің келесі мәні */
}
Printf ("\n  S=%f',s);
  /* ДЕЙІН – циклдік құрылымы, шарттың мәні ақиқат болғанға дейін
операторларды орында */
 s=0;   i=1;
 do
  s=s+(sin(i)/(2*i+1));
  i=i+1
 while  (i>n);      /*цикл соңы*/
 printf(" \n S=%f",s);
 }
 /*ҮШІН – циклдік құрылымы,  +1(-1) қадаммен операторларды
орындайды*/
s=0;
/*цикл басы,қадамы және соңы for – жолында анықталады */
for (i=1; i<=n; i++)
  s=s+(sin(i)/(2*i+1));
printf("\n S=%f",s);
getch();
}
```

Тапсырмалар

1. $\frac{1}{n+12}$
2. $\frac{n+1}{3n+3}$
3. $\frac{n^2+1}{n^3}$
4. $\frac{2n+3}{4n+5}$
5. $\frac{\sin(n)}{2n+12}$
6. $\frac{n+1}{2n}$
7. $\frac{\cos(n)}{4n+1}$
8. $\frac{arctg(n)}{5n+1}$
9. $\frac{2n^2+1}{\ln(n)}$
10. $\frac{\log(n)}{n+1}$
11. $\frac{\sin(n)}{2n+1}$
12. $\frac{2n+3}{n^3}$
13. $\frac{2n+2+\ln(n)}{n^3}$
14. $\frac{\cos(2n+1)}{n^2+1}$
15. $\frac{\sin(3n+1)}{n+1}$
16. $\frac{n^2+1}{n^3}$
17. $\frac{2n^2+1}{\ln(n)}$
18. $\frac{n+1}{2n}$
19. $\frac{2n+3}{n^3}$
20. $\frac{\sin(n)}{2n+1}$

Бақылау сұрақтары

1 Циклдік алгоритмдердің қандай құрылымын білесіз?
2 Циклдік операторлардың жұмысының принципі қандай?
3 Цикл параметрі дегеніз не?
4 Циклдік құрылымдардың негізгі элементері қандай?

Тақырып – 4 Тілдің басқарушы құрылымдары

Мақсаты: Тармақталу және қайталанбалы процесстердің күрделі алгоритмдерін бағдарламалауды үйрету

Жүйемен берілген есептердің шығарылу жолдары. Бұл есептерде нақты немесе бүтін сандарға арифметикалық операциялар қолданылады. Есептің шарты мынадай арнайы белгілермен белгіленеді:

- $\sum$ - қосынды белгісі

$$(1+1^2)+(2+2^2)+(3+3^2)+(4+4^2)+(5+5^2)$$

мына өрнектің қысқаша жазылуы:

$$\sum_{i=1}^{5} i + i^2$$

- $\prod$ - көбейтінді белгісі

$$\frac{1}{1+4} * \frac{2}{2+4} * \frac{3}{3+4}$$

мына өрнектің қысқаша жазылуы:

$$\prod_{j=1}^{3} \frac{j}{j+4}$$

- $\left\{ \right.$ - фигуралық жақша, бірнеше нұсқаларды біріктіреді, яғни берілген шартқа байланысты біреуі ғана таңдалады.

Есеп N –нің мәнін есепте

$$N = \sum_{i=1}^{5} \left(i + \prod_{j=1}^{4} \frac{i^2 + j}{j} \right)$$

Pascal – тілінде

```
Var
  n,  p : real; { n- қосынды,   p- көбейтінді }
  i, j : integer; { i,j – қос цикл параметрлері }
Begin
  n:=0;
  For i:=1 to 5 do   {сыртқы цикл}
    begin
    P:=1;
    For j:=1 to 4 do     {ішкі цикл}
      p:=p*((sqr(i)+j)/j);
    n:=n+(i+p);
    end;
 Writeln('n=',n:6:2);
End.
```

Кейде дәрежені есептеу қажет болады, бұл жағдайда стандартты фукцияларды пайдаланамыз, ол - math.h модулында анықталған:

pow(x,y) - x-тің y дәрежесі

pow10(x) – 10-нің x дәрежесі

Бұл функциялардың нәтижесі float типі болады. Олардың мәні int болу үшін үшін , оған түрлендіру операциясын қолдану қажет. Мысалы:

```
int x,y;
...
x= x+(int)pow(x,y);
```

Есеп. K- нің мәнін есепте:

$$K=\begin{cases} a+\sum\limits_{i=1}^{5} i+i^2 & ,a>0 \\ a+\prod\limits_{j=1}^{3} \frac{j}{j+4} & ,a\le 0 \end{cases}$$

C – тілінде

```
#include<stdio.h>
main()
{
float a,b ;
float k;        // нәтиже
int I,j;        // цикл параметрі
printf("\n введите a \n");
scanf("%f",&a);
if (a>b)
   {
    k=0; for(i=1; i<=5; i++) k+=i+(int)pow(i,2);
   }
else
   {
    k=1;  for(j=1; j<=3; j++) k*=(j/(j+4));
   }
K=a+k;
printf("\n k=%f",k);
}
```

Тапсырмалар

Тілдің басқарушы құрылымдарын қолданып төмендегі функциялардың мәнін есептейтін бағдарлама жазыңыз

1 N мәнін есепте:

$$N=\sum_{i=1}^{8}\left(i+\prod_{j=1}^{4}\frac{i*j}{2*i}\right)$$

2 a және b нақты сандар берілген. K мәнін есепте:

$$K = \begin{cases} a + \sum\limits_{i=1}^{5} (a * i + i^2), & a > b \\ \prod\limits_{i=1}^{4} \dfrac{a * i + b * i}{a * i + b}, & a \le b \end{cases}$$

3 K мәнін есепте:

$$K = \sum_{i=1}^{5} \sum_{j=1}^{5} (i! + i * j)$$

4 n бүтін сан берілген. K мәнін есепте:

$$K = \begin{cases} n!, & n - \text{жұп} \\ \sum\limits_{i=1}^{10} (i + i^2), & n - \text{тақ} \end{cases}$$

5 N мәнін есепте:

$$N = \prod_{i=1}^{5} \left(\sum_{j=1}^{3} (i + j) + i * \prod_{k=1}^{5} \frac{i + k}{i} \right)$$

6 a және b сандары берілген. K мәнін есепте:

$$K = \begin{cases} \sum\limits_{i=1}^{3} \sum\limits_{j=1}^{3} (a * i + b * j), & a > b \\ \prod\limits_{i=1}^{3} \sum\limits_{j=1}^{3} (a * i + a * b * j), & a = b \\ a * b + P, & a < b \end{cases}$$

мұндағы $P = \begin{cases} a, & a * b > 0 \\ b, & a * b \le 0 \end{cases}$

7 a және b натурал сандары берілген. N мәнін есепте:

$$N = \begin{cases} \sum\limits_{i=1}^{a} \prod\limits_{j=1}^{b} (a * i + b * j), & a > b \\ P + \sum\limits_{j=1}^{b} \dfrac{j}{a}, & a \le b \end{cases}$$

мұндағы $P = \begin{cases} a!, & ґ - ć \\ b!, & ґ - ў \end{cases}$

8 m бүтін сан берілген. K мәнін есепте:

$$K = \begin{cases} p + \sum_{i=1}^{3} (i + m * i^2), & m > 0 \\ m + p, & m = 0 \\ \prod_{i=1}^{3} \sum_{j=1}^{4} (i * j + m * i), & m < 0 \end{cases}$$

9 x, y, z бүтін сандар берілген. K мәнін есепте:

$$K = \begin{cases} \sum_{i=1}^{3} \frac{x * i + y * z}{i}, & x - \min \\ \sum_{i=1}^{3} \prod_{j=1}^{3} (x * i + y * j + z), & y - \min \\ x * y * z, & z - \min \end{cases}$$

10 N натурал сан берілген. K мәнін есепте:

$$K = \sum_{i=1}^{6} \left(a + \prod_{j=1}^{6} \frac{2 + j}{i} \right)$$

Бақылау сұрақтары

1 Алгоритмдердің базалық құрылымдарының қандай түрлері бар?
2 Тармақталу операторлары қандай қызмет атқарады және олардың қандай түрлері бар?
3 Қайталау операторларының түрлері қандай және олардың әрқайсысы қандай жағдайларда пайдаланылады?
4 Циклдік құрылымның қателері
5 Бос оператордың атқаратын қызметі қандай ?

Тақырып - 5 Бүтін санды арифметикамен жұмыс

Мақсаты: Натурал санның цифрларымен жұмыс істеу алгоритмдерін бағдарламалауды үйрету.

Бүтін санды арифметика, тек қана бүтін натурал сандармен орындалатын амалдар. Мұндай есептерді шығару үшін, көбіне *mod* және *div* бүтін сандық арифметкалық операциялар қолданылады.

Бұл типтегі есептерді шығару кезінде қолданылатын негізгі тәсілдер :

- натурал санның ретін анықтау;
- санның қандай да бір цифрын белгілеу;
- санның цифрларын кері орналастыру арқылы жаңа сандар алу.

Есеп: Берілген натурал санның цифрларының қосындысын және санын анықта

Берілген натурал сан: n=1324, осы санның цифрларының қосындысы S=10, ал цифрларының саны k=4. Цифрлармен жұмыс істеу үшін санның цифрларын соңынан бастап бөліп аламыз.

Ол үшін санды бүтін санды бөлу амалымен 10-ға бөліп, қалдығын аламыз: 1324 mod 10 = 4 – бұл бірінші цифр, ал келесі цифрды алу үшін бүтін санды бөлу амалымен 10-ға бөліп, санның бүтін бөлігін алу керек: 1324 div 10 = 132
Осылайша кезекпен қайталап, берілген натурал сан нолге тең болғанға дейін орындап, келесі цифрларды да алуға болады:

132 mod 10 =2 - 132 div 10 = 13
13 mod 10 =3 - 13 div 10 = 1
1 mod 10 =1 - 1 div 10 = 0

```
Var  n,a,s,k: longint;
 Begin
 Write('n=');  readln(n);
 S:=0;  k:=0;
 While n<>0 do
  Begin
   a:=n mod 10;          {санның қалдығы, n-нің  кезекті  цифры}
   S:=s+a;              {санның  цифрларының қосындысы}
   Inc(k);              {санның  цифрларының саны}
   n:=n div 10;         {санның бүтін бөлігі,  n –нің келесі мәні}
 End;
Writeln('Сумма_S=',s,'   Количество_ K=',k);
readln
End.
```

Си тілінде, / - бөлу амалы, егер айнымалылар нақты типтен болса: 5/2=2.5 ал бүтін типтен болса: 5/2=2 нәтижесі бүтін бөлігі болады, % операциясы бүтін калдықты береді: 5%2=1.

```
# include <stdio.h>
main()
{
long n;         // берілген сан
int p=0, s=0;  // нәтижесі, s - қосындысы және p- саны
printf("\n введите n\n");
scanf("%ld",&n);
while(n>0)
 {
  s+=n%10;
  n/=10;
  p++;
 }
printf("\n порядок числа p =%d",p);
printf("\n сумма цифр s=%d",s);
}
```

Тапсырмалар

1 n натурал сан берілген. Цифрларын кему реті бойынша ретте.
2 n натурал сан берілген. Санда қайталанып кездесетін цифрлар барма тексер.
3 n натурал сан берілген. Санның цифрларын кері орналастырып, жаңа сан алу керек
4 n натурал сан берілген. Цифрларының санын анықта
5 n натурал сан берілген. Клавитурадан енгізілген цифрды жойып, жаңа сан алу керек
6 n натурал сан берілген. Ең үлкен және ең кіші цифрларын тап
7 n, m натурал сандар берілген. N –нің алғашқы m цифрларының қосындысын есепте.
8 n, m натурал сандар берілген. N санынан бастап, m жай санды анықта және қосындысын есепте.
9 n натурал сан берілген. Тексер, енгізілген сан полиндром ба?
10 n натурал сан берілген. Тексер, цифрлары бірдей ме?

Бақылау сұрақтары:

1 Скалярлы типтерге қандай типтер жатады?
2 Бүтін типтердің қайсылары тек оң мәндерден тұрады?
3 Бүтін санды бөлу амалдары қандай?
4 Бүтін типті деректер туралы не білесіз?

Тақырып – 6 Ішкі бағдарламалармен жұмыс

Мақсаты: Ішкі бағдарламаларды ұйымдастыру тәсілдерін және оларды дұрыс пайдалануға үйрету

Есеп - Төмендегі функцияның мәнін ішкі бағдарламаларды қолданып есепте.

$y=\sum_{n=1}^{15}(s(n)*p(n))$, мұндағы $P(x)=\begin{cases}x^2, x<5\\ 2x, x\geq 5\end{cases}$ және $S(t)=2\cos(t)$

Паскаль тілінде

```
 var y :real;  n: integer;
function s(t:integer):real;    {ішкі бағдарлама,  s-функциясы}
 begin
  s:=2*cos(t);
 end;
function p(x:integer):integer;    {ішкі бағдарлама,  p-функциясы}
 begin
  if x>=5 then p:=2*x
     else p:=x*x;    end;
  {негізгі  бағдарлама}
begin
   y:=0;
```

```
  for n:=1 to 15 do
     {ішкі бағдарламаларға қатынау:
         s(n)-s функциясына,  p(n) - p функциясына}
  y:=y+(s(n)*p(n));
  writeln('y=', y:6:2);
readln
end.
```

Си тілінде

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float s(int t)  /*ішкі бағдарлама,  s-функциясы*/
  {
  return(2*cos(t);
  }
int p(int x)     /*ішкі бағдарлама,  p-функциясы*/
 {
 If x>=5 return(2*x);
    else return(2*x);
}
  {негізгі  бағдарлама}
void main()
 {
  float y=0;
  int n;
puts("Вычисление "); /* экранға хабарлама шығарады */
for(n=1; n<15; n++)
  y=y+(s(n)*p(n));
printf("ss=%3.2f\n",y);
printf ("Для завершения нажмите <Enter>");
getch();
}
```

Тапсырмалар

1. $y = \begin{cases} 2P(a)+S(b), a<0 \\ S(a)-P(b), a\ge 0 \end{cases}$, мұндағы $P(x) = \begin{cases} 2x, x<3 \\ x, x\ge 3 \end{cases}$ және $S(t) = \sin(2t+2)$

2. $y = \sum_{n=1}^{10}(P(n)+S(n))$, мұндағы $P(x) = \begin{cases} x^2, <5 \\ 2x, x\ge 5 \end{cases}$ және $S(t) = \cos(t^3)$

3. $y = \begin{cases} P(a)+S(b), b<0 \\ S(a)+P(b), b\ge 0 \end{cases}$, мұндағы $P(x) = \begin{cases} x^3, x>5 \\ \sin(x), x\le 5 \end{cases}$ және $S(t) = 2t+12$

4. $y = \sum_{n=1}^{25}(P(n)-S(n))$, мұндағы $P(x) = \begin{cases} \ln(x^2), x<10 \\ \log(x), x\ge 10 \end{cases}$ және $S(t)$ = жәнеrctg(t-2)

5. $y = \begin{cases} 2P(a)+S(b), a<10 \\ S(a)-3P(b), a\ge 10 \end{cases}$, мұндағы $P(x) = \sum_{n=1}^{x}\frac{1}{n}$ және $S(t) = \sin(t)-\cos(t)$

6. $y = \sum_{n=1}^{5}\frac{R(n)}{S(n)}$, мұндағы $P(x) = \begin{cases} 2x+3, x<2 \\ x^3, x\ge 2 \end{cases}$ және $S(t) = \frac{2t^2+12}{t^2+1}$

7. $y = \begin{cases} P(a) - S(b), a < b \\ S(a) - S(b), b \ge a \end{cases}$, мұндағы $P(x) = \sum_{n=1}^{x} \frac{1}{n}$ және $S(t) = \frac{12\sin(t-2)}{t^3}$

8. $y = \sum_{n=1}^{12} (P(n) + S(n))$, мұндағы $P(x) = \begin{cases} 2x, x < 6 \\ x^2, x \ge 6 \end{cases}$ және $S(t) = 12t + \sin(t)$

9. $y = \begin{cases} 2P(a) * S(b), a > b \\ S(a) - P(b), b \ge a \end{cases}$, мұндағы $P(x) = \sum_{n=1}^{x} \frac{\sin(x)}{\cos(x)}$ және $S(t) = 24t^2 - 152$

10. $y = \sum_{n=1}^{15} (S(n) - P(n))$, мұндағы $P(x) = \begin{cases} 2x + 3, x \ge 8 \\ x^3, x < 8 \end{cases}$ және $S(t) = \sin(\frac{12t}{t^2})$

Бақылау сұрақтары

1 «Функция» терминнің мағынасы қандай?
2 Функцияны анықтаудың жалпы синтаксисі қандай?
3 Функцияның формальды және фактілі параметрлерінің бір - бірінен айырмашылығы қандай?
4 Функцияны хабарлаудың ерекшеліктері қандай?
5 Функцияны шақыру үрдісін сипаттаңдар?
6 «Рекурсивті функция» терминнің мағынасы қандай?

Тақырып -7 Бір өлшемді массивтермен жұмыс

Мақсаты: *бір өлшемді сандық* массивге қолданылатын амалдарды үйрету

Есеп: Бір өлшемді массивтің кіші элементі мен үлкен элементінің орындарын ауыстыр.

```
   Var a : array[1..50] of  integer;
       i, k, l, n, max, min : integer;
Begin
 Writeln('введите размер массива ');
 Readln(n);
  {массив элементтерінің саны}
 Writeln('введите элементы массива');
 For i:=1 to n do
 Read(a[i]); {a[i]массивнің i-ші элементін клавиатурадан енгізу}
 {min - массивтің ең кіші  элементі, l - оның индексі}
    min:=a[1];    l:=1;
 {min  элементті a[i]-ші  элементпен салыстыра отырып іздеу}
   for i:=1 to  n do
   if a[i]<=min then  begin min:=a[i];  l:=i  end;

{max - массивтің ең үлкен  элементі, k - оның индексі}
 max:=a[1];  k:=1;
{үлкен элементті   a[i]-ші элементпен салыстыра отырып іздеу}
 for i:=1 to  n do
     If a[i]>=max then begin max:=a[i];  k:=i end;
```

```
  {үлкен a[k]-ді  және  кіші a[l]-дің орындарын ауыстыру}
      n:= a[l]; a[l]:=a[k];  a[k]:=n;
  {түрлендірілген  массив элементерін экранға шығару}
  For i:=1 to n do
         Write(a[i],' ');
  Writeln;
  End.
```

Есеп: Бір өлшемді массив берілген. Элементтерін кері орналастыру керек

```
#include <stdio.h>
main()
{
  int m[10], I,n,k, ind;
  puts("введите требуемую строку");
  for(i=0; i<10; i++)  scanf("%d",&m[i]);
  for(n=9; n>=0; n--)   /*n-нің мәндері 9-дан 0-ге дейін сандар*/
   { k=-32768;
     for(i=0; i<=n; i++)
        if (m[i]>k)  {k=m[i]; ind=I;}
     m[ind]=m[n]; m[n]=k;
   }
  puts("упорядоченный массив");
 for(i=0; i<10; i++)  printf("%d ",m[i]);
}
```

Массивтерді сұрыптау

Сызықты сұрыптау

Сызықты сұрыптаудың негізгі идеясы – массивтің элементтерін бірінен соң бірін тексере отырып, ең үлкен санды тауып және оны алдынғы позициядағы кіші элементпен орнын ауыстыру.

```
{массив элементтерін кему реті бойынша сызықты сұрыптау}
program Sort_lin;
const
 Count=20;
 m:array[1..Count]of byte=(9,11,12,3,19,1,5,17,10,18,3,
                                  19,17,9,12,20,20,19,2,5);
 var
  i,j,n,l: Byte;
  a:integer;
 begin
 Writeln('ishodny massiv');
 for i:=1 to count do write(' ',m[i]);
 writeln;
 a:=0;
 {Массивтің  i – ші элементін i+1 элементтен бастап соңына
дейінгі элементтермен салыстырамыз }
 for i:=1 to count-1 do
  for j:=i+1 to count do
```

```
   begin
   a:=a+1; { қайталану үрдісінің саны}
{Егер массивтің сұрыпталмаған жағында, массивтің i-ші
элементінен үлкен элемент табылса, онда олардың орындарын
ауыстыр}
   if m[i]<m[j] then
      begin
        n:=m[i];{ n - уақытша есте сақтау}
          m[i]:=m[j];
            m[j]:=n;
      end;
  for l:=1 to count do write(' ',m[l]);
  writeln('chislo iteracii',a);
  end;
  end.
```

Көпіршік әдісімен сұрыптау

Сұрыптаудың бұл әдісін кез келген массивке қолданыуға болады. Осы әдіс массивті солдан оңға қарай жүйелі түрде қарастырады, егер кезектегі элемент алдынғысынан кіші болса, сол элемент массивтегі кезекті элементпен орындарын ауыстырады. Нәтижесінде бірінші орынға ең кіші элемент шығады.

Осы үрдісті мысалмен қарастырайық:

```
program Sort_puz;{Көпіршік әдісімен кему реті бойынша сұрыптау}
const
Count=5;
m:array[1..Count]of
    byte=(9,11,12,3,19,1,5,17,10,18,3,19,17,9,12,20,20,19,2,5);
 var
  i,j,n,k,l: Byte;
  a:integer;
 begin
   Writeln('ishodny massiv');
   for i:=1 to count do write(' ',m[i]);
   writeln;
   a:=0;
   for i:=1 to count do
     begin
      for j:=count downto i do
        begin a:=a+1;
{Оң жақтағы элемент сол жақтағы элементтен үлкен болса,онда оны
солға ығыстырады}
        if m[j-1]<m[j] then
                      begin  k:=m[j-1]; m[j-1]:=m[j]; m[j]:=k;
{Әрбір алмастырудан кейін массивті экранға шығару}
                      for l:=1 to count do write(' ',m[l]);
                      writeln('chislo iteracii=',a);
                      end;
        end;
     end; {Сұрыптауды аяқтау}
end.
```

Нәтижесі

```
 9 11 12 3 19 1 5 17 10 18 3 19 17 9 12 20 20 19 2 5
 9 11 12 3 19 1 5 17 10 18 3 19 17 9 12 20 20 19 5 2   iteracia sani=1
 9 11 12 3 19 1 5 17 10 18 3 19 17 9 20 12 20 19 5 2    iteracia sani=5
  ... ... ...
 20 20 19 19 19 18 17 17 12 12 11 10 9 9 5 3 5 3 1 2    iteracia sani=164
 20 20 19 19 19 18 17 17 12 12 11 10 9 9 5 5 3 3 2 1     iteracia sani=170
```

Тез сұрыптау әдісі(Хоардың сұрыптау әдісі)

Сызықты қарапайым орын ауыстыру, «көпіршік» әдістерімен сұрыптау тиімсіз болып табылады. Бұл сұрыптау үрдісінде тек қана өзара көршілес элементтерді салыстыру мен орын ауыстыруды қажет ететін әдіс идеясы берілген.

Сұрыптау әдісін орын ауыстыру негізінде айтарлықтай жақсартуға болады. Оған бүгінгі күні бөлу арқылы орын ауыстыру деп аталатын сұрыптау әдісін қолдануға болады. Ол бір-бірінен үлкен арақашықтықта тұрған элементтерді салыстыру мен орын ауыстыруға негізделген. Бұл әдісті 1962 жылы Ч.А.Р.Хоар ұсынған. Осы әдістің өнімділігі көңіл толарлықтай болғандықтан, автор оны «тез сұрыптау» деп атады.

```
{Массивті бөліктерге бөлу арқылы тез сұрыптау  }
program quck_sort;
uses crt;
const count=20;
m: array[1..count] of
     byte=(9,11,12,3,19,1,5,17,10,18,3,19,17,9,12,20,20,19,2,5);
var
i,a:integer;
procedure quicks(first,last:integer);
var i,j,x,w,l:integer;
begin
 i:=first;{Массивтің сол жақ шекарасы - 1 элемент}
 j:=last;{ Массивтің оң жақ шекарасы - 2 элемент }
 x:=m[(first+last) div 2];{Массивтің ортасын анықтау}
 repeat
  while m[i]>x do i:=i+1;
  while x>m[j] do j:=j-1;
  a:=a+1;{Элементтердің рын ауыстыру итерациясының саны}
  if i<=j then
              begin {Элементтердің орнын ауыстыру}
               w:=m[i]; m[i]:=m[j]; m[j]:=w;
               i:=i+1;
               j:=j-1;
   {Әрбір ауыстырудан кейін экранға массивты шығару}
               for L:=1 to count do  write(' ',m[l]);
               writeln('chilso iteracii=',a);
              end;
 until i>j;
    {quicks процедурасын рекурсиялық шақыру }
```

```
   if first<j tneh quicks(first,j);
   if i<last then quicks(i,last)
end;{Сұрыптаудың соңы}
  {Негізгі бағдарламаның басы}
   begin  clrscr;
   writeln('ishodny massiv:');
   for i:=1 to count do write(m[i]:3,' ');
   writeln;
   a:=0;
   quicks(1,count);{ quicks сұрыптау  процедурасын шақыру}
   writeln('otsortirovany massiv:');
   for i:=1 to count do write (m[i]:3,' ');
   writeln;
   end.
```

Нәтижесі

9 11 12 3 19 1 5 17 10 18 3 19 17 9 12 20 20 19 2 5
19 11 12 3 19 1 5 17 10 18 3 19 17 9 12 20 20 9 2 5 iteracia sani=1
19 20 12 3 19 1 5 17 10 18 3 19 17 9 12 20 11 9 2 5 iteracia sani=2
19 20 20 3 19 1 5 17 10 18 3 19 17 9 12 12 11 9 2 5 iteracia sani=3
… … …

20 20 19 19 19 18 17 17 12 12 11 10 9 9 5 5 3 1 2 3 iteracia sani=26
20 20 19 19 19 18 17 17 12 12 11 10 9 9 5 5 3 3 2 1 iteracia sani=27
20 20 19 19 19 18 17 17 12 12 11 10 9 9 5 5 3 3 2 1 iteracia sani=28

Берілген мысалдардан “тез” сұрыптау алгоритмі, көпіршік, сызықты сұрыптау әдістеріне қарағанда жақсы нәтиже береді, бірақ кейбір жағдайда кемшілігіде болады.

Жоғарыдағы мысалдарға, берілген массив элементтеріне назар салсаныз байқайтыныңыз: тез сұрыптау 28 итерацияда орындалса, сызықтық әдіспен 190 итерация, ал көпіршік әдісімен 166 итерацияда орындалды. Бұл жағдайлар алгоритмнің тиімділігін анықтауға мүмкіндік береді.

Тапсырмалар

1 N өлшемді A бүтін сандар сызықты массиві берілген. Он элементтерінің санын анықтайтын бағдарлама құрастыр.

2 N өлшемді A бүтін сандар сызықты массиві берілген. Тақ номерлі элементтердің қосындысын есептейтін бағдарлама құрастыр.

3 N өлшемді A бүтін сандар сызықты массиві берілген. Элементтерін кему реті бойынша реттейтін бағдарлама құрастыр.

4 Сызықты массив берілген. Осы массивтің ең үлкен элементін және оның индексін табу керек.

5 Сызықты массив берілген. Массивтің оң элементтерінің қосындысын, теріс элементердің модульдерінің қосындысын және санын табу керек.

6 A(a_1,a_2,…,a_{25}) сызықты массив берілген. $|a_i|>0$ шартын қанағаттандыратын барлық элементтердің арифметикалық ортасын анықта.
7 Сызықты массив берілген. Осы массивтің ең үлкен элементі мен ең кіші элементінің орындарын ауыстыр.
8 Сызықты массив берілген. Осы массивте қайталанып кездесетін элементтер бар ма, тексер.
9 Сызықты массив берілген. Осы массивтің элементтерін кері орналастырып, оларды кему реті бойынша ретте.
10 Сызықты массив берілген. Осы массивтің жұп элементтерінің санын анықта.

Бақылау сұрақтары
1 «Массив», «индекс» терминдерінің мағынасы қандай?
2 Массивтің соңғы элементінің индексі қандай болады?
3 Массивтің элементтері бағдарламада қалай жазылады?
4 Массивтің элементтерінің типі қандай болуы мүмкін?
5 Массивтермен жұмыс істеу негізгі операциялары қандай?

Тақырып – 8 Екі өлшемді массивтермен жұмыс

Мақсаты: *екі өлшемді сандық* массивтерге қолданылатын амалдарды үйрету

Есеп: Матрица бағандарының элементтерінің қосындысын есепте.

```
   var  a : array[1..10,1..10] of integer;
        s : array[1..10] of integer; {бағандарының қосындысы}
        i, j, n, m : integer;
begin
  randomize;
  writeln('введите размер матрицы  n*m :');
  readln(n,m);   {n- жолдарының саны, m –бағандарының саны}
  for i:=1 to n do
    begin
   for j:=1 to m do
   begin
{элементтерді кездейсоқ  сан функциясының көмегімен анықтайды}
  a[i,j]:=random(100);
{элементтерді матрица түрінде шығару}
  write(a[i,j], '  ')
  end; writeln
  end;
    { әрбір  баған элементтерінің қосындысы нәтижесінде  S - бір
                  өлшемді массивті қалыптастырады}
   for j:=1 to m do
    begin
    s[j]:=0;
   for i:=1 to n do
```

```
  s[j]:=s[j]+a[i,j];  {s[j] – баған элементтерінің қосындысы }
     end;
     for j:=1 to m do   write(s[j],'  ');
     writeln;
readln  end.
```

Есеп: n*m өлшемді екі өлшемді массив берілген.Жолдар мен бағаналардың қиылысуындағы орналасқан минималды элементінің нөмірін табыңыз .

```
# include <stdio.h>
main()
{
int n,m,a[10][10]; // алғашқы массив
int k;    // жол нөмірі
int pr; // pr – шарты белгі: 0 – жол табылған жоқ, 1 - табылды
int s;          // Жолдың теріс элементтерінің саны
int i,j,x;
printf("\n n  және m массивтердің өлшемін енгізіңіз ");
scanf("%d%d",&n,&m);
printf("\n жол бойынша массивтің элементтерін енгізіңіз ");
for(i=0;i<n;i++)
 for (j=0;j<m;j++)
   scanf("%d",&a[i][j]);
pr=0;
for (i=0; i<n && !pr; i++)
    {
      s=0;
      for (j=0; j<m;j++)
      if (a[i][j]<0)  s++;
      if (s==m)  { k=i; pr=1;}
     }
for (j=0;j<m;j++)
     {
       x=a[0][j]; a[0][j]=a[k][j]; a[k][j]=x;
     }
// нәтижесі
for(i=0;i<n;i++)
 {
  for (j=0;j<m;j++)
  printf("%d  ",a[i][j]);
  printf("\n");
 }
}
```

Тапсырмалар

1 A(n,m) матрицасындағы оң және теріс элементтер санын анықта.

2 n*m өлшемді матрица берілген. Әрбір жолының элементтерінің қосындысын есепте.

3 n*m өлшемді матрица берілген. Бас диагоналындағы элементтердің ішінен үлкен элементті табу керек.

4 n*m өлшемді матрица берілген. Қайталанып кездесетін элементтер бар ма?

5 n*n өлшемді А матрицасының ең үлкен элементінің координатын (жолымен баған нөмірін) анықта.
6 В матрицасының бас диагоналынан жоғары орналасқан элементтерінің қосындысын есепте.
7 n*m өлшемді матрицаның кіші элементі мен үлкен элементінің орындарын ауыстыр.
8 j*m және n*i өлшемді А және В матрицалары берілген. Матрицалардың көбейтіндісін табу керек.
9 A(n,n) матрицасының бас диоганалындағы ең үлкен элементін тауып, сол элемент тұрған жолды экранға шығар.
10 n*m екі өлшемді массив берілген. Белгілі шарт бойынша массивтің жолдарын ауыстыру керек.

Бақылау сұрақтары:
1 Екі өлшемді массив дегеніміз не?
2 Массивтің базалық типтері қандай ?
3 Массив индексі қандай тип бола алмайды?
4 Массвтермен қандай операциялар орындалады?
5 Массивтерді сұрыптау әдістеріне негізгі талап қандай?

Тақырып - 9 Жолдармен жұмыс
Мақсаты: символдық шамалармен таныстырып, олармен орындалатын амалдарды дұрыс пайдалануға үйрету

Есеп: Жол берілген. Жолдағы барлық «А» әріптерін жолдың басына орналастыр.

```
 Var  s, s1: string;
         i: integer;
begin
  Writeln('введите строку');
  Readln(s); { s - берілген  жол, клавиатурадан енгізіледі}
    s1:=''; {бос жол}
 { жолдағы символдарды ізделінді символмен салыстырып, қажетті жаңа
                          жол қалыптастыру }
 For i := 1 to length(s) do
    If s[i] = 'A' then  s1 := s[i] + s1
                      else s1 := s1 + s[i];
   Writeln(s1);
    readln
end.
```

Есеп: Бұл бағдарлама енгізілген символдарды керісінше шығарады. Мысалы, егер 0123456789 жолын енгізсек, онда ол былай өзгереді: 9876543210.

```
  #include<stdio.h>
  #include<string.h>
```

```
  main()
  {  char s[10];  /* символдық типті s массивін хабарлау */
  int i,
  puts("введите строку из десяти символов);
   for(i=0; i<=10; i++)
     scanf("%c",&s[i]) /*элементтерді енгізу*/
  for(i=9; i>=0; i-- )
    print("%c",s[i]); /* элементтерді кері шығару*/
```

Есеп: Сөздері бос орынмен бөлінген жол берілген. Сиволдарының саны жұп болатын барлық сөздерді жою керек.

```
# include<stdio.h>
# include<string.h>
main()
{
char a[30];    // бастапқы жол
int i,j;
printf("\n жолды енгізіңіз \n");
gets(a);
strcat(a," ");       // а жолының соңына бос орынды қосу
i=0; j=0;
while(j<strlen(a))
  if (a[j]==' ')
    { if ((j-i)%2==0)
       { strcpy(a+i,a+j+1); j=i; }
       else
        {j++; i=j; }
    }
  else
      j++;
printf("\n %s",a);
}
```

Тапсырмалар

1 Символдық жол берілген. Берілген жолдағы «авс» тіркесінің санын анықтайтын бағдарлама жаз.

2 Символдық жол берілген Берілген жол керісінше оқығанда да сол мағынаны білдіреме анықта, мысалы-қазақ, шалаш, 2002 .

3 Символдық жол берілген. Берілген жолды кері оқитын бағдарлама жаз

4 Символдық жол берілген. Жолдағы А әріпіне аяқталатын ең үлкен сөзді анықта.

5 Символдық жол берілген. Жолдағы а әріпінен басталатын сөздерді экранға шығар.

6 Символдық жол берілген. Жолдағы кездескен нүктелерді сұрақ белгісімен алмастыр.

7 Символдық жол берілген. Жолдағы кездескен бос орындарды(пробел) жою керек.

8 Символдық жол берілген. Жолда қайталанып кездесетін символдар бар ма, анықта.

9 Символдық жол берілген. Жолдың символдарының саны жұп болатын сөздерін экранға шығар
10 Символдық жол берілген. Жолдың ең кіші сөзін жолдың соңғы сөзімен алмастыр

Бақылау сұрақтары
1 Жолдық айнымалылар қалай сипатталады?
2 Бір жолдық айнымалыға немесе тұрақтыға қанша символ жазуға болады?
3 Жолдық айнымалының ұзындығы қалай анықталады?
4 Жолдарды өңдеу фнкуциялар кітапханасы қандай файлда орналасқан?
5 Жолдарды салыстыру үшін қандай функцияларды қолдануға болады?
6 Жолдарды қосу және көшіру үшін қандай функцияларды қолдануға болады?

Тақырып -10 Жазба және файлмен жұмыс
Мақсаты: Қарапайым деректер базасының файлдық құрылымын жасауды үйрету

Есеп. Теріс және он бүтін сандардан тұратын f файлы берілген. G – файлына он сандарды жазу керек.

```
var
    i : integer;
begin
   assign(input,'f.txt');
assign(output,'g.txt');
  reset(input);   rewrite(output);
   while  not(eof(output))  do
     begin
       read(i);
          if i>0 then    write(i);
      end;
   close(output);
  close(input);
   end.
```

Есеп: «Телефон анықтамасы» деректер базасын(ДБ) жасау керек. Бағдарлама ДБ-сына жаңа жазба енгізу және іздеу операцияларын орындайды. Деректер TEL.DAT. файлында сақталады

```
uses crt;
type BD = record
{BD – типті жазбаның  құрылымы}
     fio: string;
     tel: string [8];
     adres: string;
    end;
var f   : file of  BD;
{ f- файлдық айнымалы, ДБ  файлдық құрылымы}
   x  : BD;
   name : string;
   n   : integer;
             {ДБ-сына  жаңа жазба енгізу процедурасы}
procedure wwod;
begin
   rewrite(f);   { деректерді  жазу үшін файлды ашу  процедурасы}
   writeln('Введите  новые  данные  (конец  ввода  -  ввод  пустой
записи');
   repeat
   write('Введите фамилию  '); readln(x.fio);
   write('Введите телефон  '); readln(x.tel);
   write('Введите адрес    '); readln(x.adres);
{тексеру, егер  жазба бос  болмас онда деректер файлға жазылады}
   if   x.fio<>'' then write(f,x)
   until   x.fio='';
   close(f)  {файлды жабу}
end;
      {ДБ-сынан  фамилия бойынша жазба іздеу  процедурасы}
procedure poisk;
begin
    reset(f);  { деректерді оқу үшін файлды ашу  процедурасы}
write('Введите фамилию или начальные буквы для поиска');
readln(name);
   {файлдың соңы болғанға дейін, операторларды орында }
 while not(eof(f)) do
   begin
   read(f,x);  { Файлдан жазбаларды оқиды}
   if name=copy(x.fio,1,length(name))
     then begin writeln('Фамилия  ',x.fio);{ экранға шығару}
   writeln('Телефон  ',x.tel);
   writeln('Адрес    ',x.adres)
end;
 end;
close(f);
  readkey
end ;
                        {негізгі бағдарлама }
begin
  assign(f, 'TEL.DAT');   { 'TEL.DAT' - файлымен байланыс}
repeat
   clrscr;
   writeln('1.Поиск.');
   writeln('2.Ввод новых записей.');
```

```
   writeln('3.Конец работы.');
   writeln;
    writeln('Введите номер пункта'); readln(n);
  { таңдау операторы,  n –айнымалысының мәніне байланысты қажетті
                       процедура орындалады}
  case n of
    1 : poisk;
    2 : wwod ;
    3 : halt
  end;
until false
end.
```

Есеп: Аты мен жасы көрсетілген қызметкерлер тізімі берілген. 60 жасқа келген қызметкерлерді тізімнен шығарып, қалғандарын басылымға жіберу керек.

```
#include <stdio.h>
main()
{
 Struct   // Си тіліндегі жазба құрылымы
 {
  char name [20];
  int age; }
   a[10];   // қызметкерлер массиві
  int n;   //  массив өлшемі
  int i, j;
printf("\n  өлшемін енгізіңіз \n");
scanf("%d",&n);
 for (i=0; i<n; i++)
  {
   printf("\n %d –ші қызметкердің фамилиясы",i+1);
   scanf("%s",a[i].name);
   printf("\n жасы");
   scanf("%d", &a[i].age);
  }
i=0;
while (i<n)
 if (a[i].age>60)
   { for (j=i+1; j<n; j++)
          a[j-1]=a[j];  n--; }
 else
   i++;
for (i=0; i<n; i++)
 printf("\n %s  %d", a[i].name, a[i].age);
 }
```

Есеп: Әр элементі абоненттің аты-жөнін, адресін, телефонын қамтитын файл берілген. Телефон станциясымен көрсетілген абоненттердің мәліметтері бар элементтерден жаңа файл құру керек

Fscanf және fprintf функцияларының көмегімен файлға жазу және файлдан оқу операциялары орындалады.

Келесі құрылым арқылы, есептегі берілгендерді анықтайық:

```
struct abonent {
      char name  [20];
      char adress[40];
      char phone [8];
         };
```

Айнымалыны жариялаймыз:

```
      struct abonent x;
```

Төменде x айнымалысы мен құрылым элементерін байланысын көреміз:

```
      printf("\n аты-жөні");
      scanf("%s",x.name);
      printf("\n адресі");
      scanf("%s",x.adress);
      printf("\n телефон");
      scanf("%s",x.phone);
```

Толтырылған құрылымды толығымен fwrite функцясы арқылы файлға енгіземіз:

```
      fwrite(&x,sizeof(abonent),1,a);
```

Енді жоғарыдағылардың бәрін байланыстырамыз:

```
#include <stdio.h>
main()
{
FILE *a;
struct abonent {
  char name  [20];
  char adress[40];
 char phone [8];
   };
struct abonent x;
int n;
a=fopen("ff","w");
for (i=0; i<n; i++)
   {
     printf("\n аты-жөні");
     scanf("%s",x.name);
     printf("\n адресі");
     scanf("%s",x.adress);
     printf("\n телефон");
     scanf("%s",x.phone);
     fwrite(&x,sizeof(abonent),1,a);
     }
fclose(a);
}
```

Негізгі бағдарлама, fread функциясы көмегімен берілген файлдағы барлық элементтерді оқу керек, ал шартты қанағаттандырған элементтерді шығу файлына жазу керек. Бұл бағдарлама алдындағы мысалдың негізгі бағдарламасымен ұқсас болады.

```
 #include <stdio.h>
main()
{
FILE *a,*b;
struct abonent {
 char name  [20];
 char adress [40];
 char phone [8]; }
struct abonent x;
char stantion;
printf("\n станцияның нөмірін енгізу керек");
scanf("%c",&stantion);
a=fopen("ff","r");
b=fopen("gg","w");
fread(&x,sizeof(abonent),1,a);
while (!feof(a))
  {
 if (x.phone[0]==stantion)
         fwrite(&x,sizeof(abonent),1,b);
  fread(&x,sizeof(abonent),1,a);
  }
fclose(b);
b=fopen("gg","r");
fread(&x,sizeof(abonent),1,b);
while (!feof(b))
 {
 printf("%s  %s  %s \n",x.name, x.adress, x.phone);
 fread(&x,sizeof(abonent),1,b);
 }
 fcloseall();
}
```

Тапсырмалар

1 Емтихан қортындысы туралы мәліметтер: емтихан тапсырушылар тізімі және үш пәннің бағалары берілген. Білім сапасын анықтайтын бағдарлама құрастыр. Деректер файлда сақталып және файлдан оқылуы керек.

2 Жолаушылардың қол жүгі туралы мәліметтер: фамилиясы және қол жүгінің салмағы берілген. Экранға тексеру пунктінен өткізу немесе өткізбеу туралы хабарламаны шығару керек. Бір адамға 36 кг ғана алып өтуге рұқсат етіледі. Деректер файлда сақталып және файлдан оқылуы керек.

3 Автокөлік жүргізушілері туралы мәліметтер: фамилиясы, автокөлік маркасы, нөмірі берілген. Нөмір арқылы автокөлік маркасын және фамилиясын анықта. Деректер файлда сақталып және файлдан оқылуы керек.

4 Кітаптар картотекасында – автор, кітаптың аты, шыққан жылы, баспасы берілген. Авторларды алфавит бойынша реттеп, көрсетілген жылда шықан кітаптар тізімін экранға шығар. Деректер файлда сақталып және файлдан оқылуы керек.

5 Спорттық ойындардың қорытындысын шағаратын бағдарлама жазу керек. Бағдарлама қолданушыдан әр команда сайыскерлері ұтып алған әр түрлі

медальдар санын, жалпы медальдардың санын және соған сәйкес келетін бағалар санын есептеу керек, содан кейін ғана алынған бағалар саны бойынша тізімді келтіру керек. Бағалар саны келесі ереже бойынша есептеледі: алтын медальға команда жеті деген бағаны алды, күміске – алты, қолаға – бес деген баға. Деректер файлда сақталып және файлдан оқылуы керек.

6 Жолаушылар туралы мәліметтер: фамилиясы, жүктерінің саны және салмағы берілген. Экранға жүктерінің саны және салмақтары бірдей жолаушылардың тізімін шығару керек. Деректер файлда сақталып және файлдан оқылуы керек.

7 Оқушылар туралы мәліметтер: фамилия мен аты және қай сыныпта оқитындығы туралы. Фамилиялары бірдей сыныптас оқушылар бар ма, анықта. Деректер файлда сақталып және файлдан оқылуы керек.

8 Ойыншық дүкені: ойыншықтардың атаулары және бағасы, саны берілген. Ең қымбат ойыншықтар саны қанша, анықта. Деректер файлда сақталып және файлдан оқылуы керек.

9 Топтағы студенттер туралы мәліметтер: ФАЖ, туған жылы, туған жері. Туған жылдарының кему реті бойынша студенттер тізімін шығар. Деректер файлда сақталып және файлдан оқылуы керек.

10 Студенттер туралы мәліметтер: ФАЖ, бес пән атаулары және бағалары. Барлық пәндер бойынша аттестациядан өтпеген студенттер тізімін анықта. Деректер файлда сақталып және файлдан оқылуы керек.

Бақылау сұрақтары

1 «Құрылым» және «Жазба» терминдерінің мағынасы қандай?

2 Құрылымды хабарлау мен инициализациясы қалай жүргізіледі?

3 Құрылымдар массивтері қалай ұйымдастырылған?

4 Файлдық тип дегеніміз не?

5 Файл элементтеріне тура және тізбектей жетуі бір бірінен немен ажыратылады?

Тақырып -11 Мәтіндік режімде жұмыс

Мақсаты: Паскаль тілінде экранды мәтіндік режімде басқаратын CRT модулінің процедураларымен және функцияларымен жұмыс істеуді үйрету

Турбо Паскальда пернетақтаны, экранды, дыбыстарды басқаратын бірнеше стандартты функциялары мен процедуралары бар, олар кітапханалық CRT (Cathode ray Tube Display – электрондық сәулелік түтікшедегі дисплей) модулінде жинақталған.

Мәтіндік режімде экран 40 немесе 80 жолдан тұратын мәтін шығара алады. Мәтіндік режімде процедураларды пайдалану үшін алдымен CRT модулін іске қосу керек.

Бағдарламаның басында:

Паскаль Си

uses crt; #include<conio.h>

Мәтіндік режімді басқару модулін пайдалана отырып, экранға шығарылатын символдардың түсін, мөлшерін, экранның түсін де өзгертуге болады. Сонымен бірге экранда терезе жасауға, оны өшіруге, курсорды автоматты түрде жаңа орынға көшіруге, оның жаңа орнын анықтауға, жолдарды өшіруге және жалғастыруға т.б. көптеген әрекеттер істеу мүмкіндігі бар. Олар орындайтын бағдарламалар процедура түрінде берілген.

Біз олардың негізгілерін қарастырайық. Экранға шығарылатын символадар түсін **Textcolor(Color)** процедурасы арқылы өзгертеді, мұндағы Color – стандарты түс нөмірі немесе атауы(Кесте - 1).
Экранның түсін өзгерту үшін **TextBackground(Color)** процедурасы пайдаланылады.

Кесте 1.

Түс атаулары және олардың нөмірлері

Түс атауы	Нөмірі	Түс атауы	Нөмірі
Black (қара)	0	DarkGray (сұр)	8
Blue (көк)	1	LightGreen (көкшіл)	9
Green (жасыл)	2	Light Green (ақшыл жасыл)	10
Cyan (көгілдір)	3	Light Cyan (ақшыл көк)	11
Red (қызыл)	4	Light Red (қызғылт)	12
Magenta (күлгін)	5	Light Magenta (шық күлгін)	13
Brown (қоңыр)	6	Yellow (сары)	14
Lightgray (боз)	7	White (ақ)	15
		Blink (жылпылықтау)	16

Clrscr – экранды немесе терезені тазалап, курсорды экранның сол жақ жоғарғы бұрышына көшіреді, тек мәтіндік режімде жұмыс істейді.
GotoXY (X,Y) - курсор координаталарын экранның X,Y позициясына көшіреді. X сол жақ шеттен оң жаққа қарай, ал Y жоғарыдан төмен қарай берілген қашықтықтар бірлігін көрсетеді. Егер экран толық ашылған терезе деп қарастырылса, онда 25 жолдан және әр жол 80 символдан тұратын дисплей адаптерінің мәтіндік режімінде X = 1..80, Y = 1..25 мәндеріне сәйкес келеді. Экранның сол жақ жоғарғы бұрышы координаталары – (1,1), оң жақ төменгі бұрыш үшін – (25,80) болып келеді.
Windows (X1,Y2,X2,Y2) - экранда терезе құру процедурасы, X1,Y1, X2,Y2 – терезе төртбұрышының сол жақ жоғарғы және оң жақ төменгі төбелерінің координаталары. Ең кіші терезе бір символды ғана қоршаса, ең үлкен терезе (1,1,80,25) мөлшерінді бүкіл экранды алып тұрады.

Есеп, Экранға ASCII–кестесіндегі символды және оның кодын экранға шығаратын бағдарлама құрылымы.

Паскаль тілінде

```
   uses crt;
begin
 textbackground(1); clrscr; {экран фонын орнату,  1-көк түс}
       {терезе орнату процедуралары, 2- жасыл түс }
  textbackground(2);
   window(4,1,59,2);  clrscr;
  textbackground(3);
    {шығарылатын символдың позициясын орнатады, 6-шы баған, 4-ші
жол}
  gotoxy(6,4);
   write('  символ и  код символа ');
 {chr(код) – символ және  ord('символ') -кодын шығарады  }
  writeln(chr(65), ' - ', ord('A');
end.
```

Есеп , экранға мәтіндік режімде тіктөртбұрыш шығарады.

Си тілінде

```
#include <stdio.h>
#include <conio.h>
 //рамкалар сызаатын функция
   void fname (int l, int t, int w, int h)
   {
     /*l,t – жоғарғы сол бұрыштың координаталары*/
     //w,h – рамканың ені мен биіктігі
    int x,y;   /*шығарылатын символдың координаталары*/
    int i;
     /*рамканы құрайтын символдар*/
     char c1 = 218,      /*жоғарғы сол бұрыш*/
          c2 = 196,      /*көлденең сызығы*/
          c3 = 191,      /*жоғарғы оң бұрыш*/
          c4 = 179,      /*тігінен сызығы*/
          c5 = 192,      /*төменгі сол бұрыш*/
          c6 = 217,      /*төменгі оң бұрыш*/
     gotoxy (l,t);
     putch (c1);
     for (i=0; i<w-2; i++)    /*рамканың жоғарғы шегінің символы*/
          putch(c2);
     putch(c3);
      y=t+1;
      x=l+w-1;
     for (i=0; i<h-1; i++)    /*сол және оң шегінің символдары*/
     {
          gotoxy(l,y);
          putch(c4);
          gotoxy(x,y);
          putch(c4);
          y++;
     }
     gotoxy(l,y);
```

```
     putch(c5);
     for (i=0; i<w-2; i++)    /*төменгі шектің символдары*/
          putch(c2);
          putch(c6);
     }
void main()
{
clrscr();
frame(5,5,30,10);
puts("\n Для завершения нажмите <Enter>");
getch();
}
```

Тапсырмалар

1 Экранға түрлі – түсті мәтінді шығару бағдарламасын құрастыр.
2 Бағдарламаның алғашқы бетін(заставкасын) жазу керек, онда: ТАЖ(ФИО), бағдарламаның аты болу керек.
3 ASCII – кодтар кестесіндегі псевдографиканы пайдаланып суреттер сал.
4 Экранда мәтіндік режімде ақпаратты терезелерге орналастыр.

Бақылау сұрақтары

1 Курсорды енгізу процедурасы
2 Символдар түсін бояу процедурасы
3 Input буферіндегі бір символды есептейтін функция
4 Экранмен жұмыс істеу кезінде қолданылатын процедура

Тақырып – 12. Графикалық режимде жұмыс

Мақсаты: Паскаль тілінде графикалық режімде қолданылатын процедуралармен және функциялармен жұмыс істеуді үйрету.

Графикалық операторлармен жұмыс істеу үшін ең бірінші графиканы инициализациялау керек. Мәтіндік режімге өту үшін графикалық режімді жабу керек. Графиканы инициализациалайтын және жұмысының соңында мәтіндік режімге өтетін бағдарламаның фрагментін қарастырайық. Берілген фрагментте бағдарлама автоматты түрде өзі адаптердің түрін және қолданылатын графикалық режімді тексереді. Төмендегі графикалық режімнің негізгі тұрақтылары берілген.

Паскаль

```
uses Graph;
 var
  Gd, Gm : Integer;
begin
Gd := Detect;
InitGraph(Gd, Gm,' ');
if GraphResult <> grOk then
Halt(1);
```

Си

```
#include <graphics.h>
#include <stdio.h>
main()
{
int gd = DETECT, gm, errorcode;
initgraph(&gd, &gm, "");
if (errorcode != grOk)
  {
   printf("Graphics error:",
```

```
{бағдарламаның графикалық
бөлімі)
ReadLn;
CloseGraph;
end.
```

```
    exit(1);
    }
{бағдарламаның графикалық
бөлімі)
getch();
closegraph();
}
```

Графикалық режімнің кейбір функцияларымен және процедураларымен, айнымалыларымен және тұрақтыларымен танысайық.

SETCOLOR(C)- түсін орнату.

PutPixel (X,Y,C)- С түсімен Х,У графикалық координаталармен нүктені салады.

Line (X1,Y1,X2,Y2)- [X1,Y1] және [X2,Y2] координатасымен кесіндіні салады.

LineTo (x, y) – x,y координаталарымен берілген нүктелер арқылы кесінді салады.

LineRel (dx, dy) – Нүктенің соңынан dx, dy векторымен бұл нүктеден кесіндіні өткізеді. **MoveRel (dx, dy)** – векторға графикалық курсорды ығыстыру(экранда із қалмайды).

MoveTo (x, y) - x, y нүктелеріне ығысу (экранда із қалмайды).

RECTANGLE(X1,Y1,X2,Y2,A)- [X1,Y1], [X2,Y2]диагонал соңдары көмегімен төртбұрыш салу.

OutTextXY (x, y, Text) – x,y графикалық координталары бар нүктеден бастап, графикалық экранда мәтінді шығару.

Bar (x1, y1, x2, y2) – Оюмен және түспен боялған тікбұрышты салады.

Bar3D (x1, y1, x2, y2, h, log) – Оюмен және түспен боялған параллелипипедті салады.

Circle (x, y, r) - **x,y** координаталарымен және r радиуспен шеңберді салу.

SetFillStyle (st, z) – фигураны стандартты боямен және түспен толтыру, егер st – сызық стилі (кестені қара), z – толтыру түсі. (Кесте 2)

SetTextStyle (Font, N, R) - Берілеген шрифтің типін, бағытын (N), символдың өлшемін анықтайды.

InitGraph (gd, dm, path) - Графикалық режімнің инициализациясы, мұндағы gd – графикалыцқ драйвердің коды, gm – графикалық режімнің коды, path – дискідегі графикалық драйверге жол.

Detect –ЭЕМ – де орнатылған графикалық драйвердің типін және графикалық режим кодын қайтаратын функция. Жоғарыда көрсетілген мысалды қара.

DetectGraph (gd, gm) – Алдындағы функцияға ұқсас процедура.

CloseGraph – Графикалық режим жұмысын аяқтайтын және ЭЕМ – ді мәтіндік режімге ауыстыру процедурасы.

ClearDevice – Графикалық экранды тазарту процедурасы.

SetBkColor (z) – Экранның фон түсін орнату.

SetColor (z) - Фигуралардың және мәтіннің түсін орнату.

SetLineStyle (st, ch, t) – Әр түрлі фигуралар көрсететін сызықтардың параметрлерін береді.

Есеп. Саңырауқұлақ пен айдың суретін салатын бағдарлама.
Паскаль тілінде

```
 Uses Graph, Crt;
Var GraphDriver, Graphmode: integer;
Begin
Clrscr;
GraphDriver:=0;
InitGraph(GraphDriver, GraphMode, ' ');
If GraphResult <> 0 then Halt;

SetBkColor(4); Setcolor(1);
Pieslice(350, 150,0,180,150);
 Setfillstyle(1,1);
  Floodfill(350,150,1);
SETcolor(5);
 Rectangle(305,150,375,350);
  Setfillstyle(1,5);
   Floodfill(308,160,5);
SETcolor(14);
  Arc(6,150,300,60,147);
    arc(50,150,280,79,130);
     Setfillstyle(1,14);
       Floodfill(167,150,14);
Readln; closegraph; End.
```

Кесте 2. Графикалық режімнің тұрақтылары

Сызық түрлері		
Тұрақты	*Оның мәні*	*Сипаттамасы*
SolidLn	0	Үздіксіз сызық
DottedLn	1	Нүктелерден тұратын сызық
CtnterLn	2	Нүктелер мен сызықшадан тұратын сызық
DashedLn	3	Штрихті пунктирлі сызық
UserBitLn	4	Орындаушы өзі тағайындайтын сызық типі

Стандартты толтыру стилдері	
Тұрақты	*Толтыру үлгісі*
EmptyFill	Экран түсімен толтырады
SolidFill	Ағымдағы түспен толтырады
LineFill	Символдармен, Color түсімен толтырады
LtslashFill	Қалыптағы символдармен, Color түсімен толтырады
BkslashFill	Қалыңдатылған символдармен, Color түсімен толтырады
LtbkSlashFill	Қалыңдатылған символдармен, Color түсімен толтырады
HatchFill	Қалыптағы символдармен, Color түсімен толтырады
XhatchFill	Тік және көлденең штрих сызықтармен, Color түсімен толтырады
InterLeaveFill	Екі диагональ бойынша қиғаш бағыттағы сиретілген сызықтармен, Color түсімен толтырады

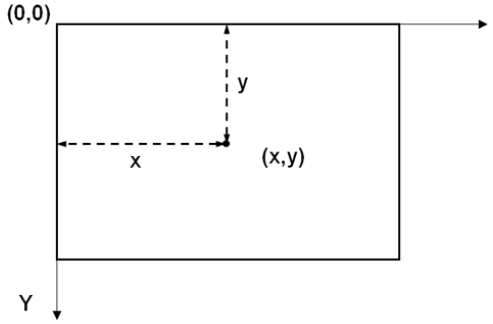
WideDotFill	Сиретілген нүктелермен, Color түсімен толтырады
CloseDotFill	Жиі орналасқан нүктелермен, Color түсімен толтырады
UserFill	Орындаушы өзі анықтаған үлгімен, Color түсімен толтырады

Есеп. Экранға қарапайым фигураларды шығару

Си тілінде

```
 #include<stdio.h>
#include<conio.h>
#include <graphics.h>
main( )
{ int gd=DETECT,gm,a=1;
initgraph(&gd,&gm,"");
circle(100,120,80);
 ellipse(100,120,0,359,80,30);
  arc(200,150,0,90,90);
  line(200,60,290,150);
  setfillstyle(7,1);
    pieslice(100,300,0,135,70);
  setfillstyle(4,1);
    pieslice(100,300,135,225,70);
  setfillstyle(10,1);
    pieslice(100,300,225,360,70);
  setfillstyle(1,1);
    bar(200,250,280,400);
  setfillstyle(1,2);
     bar3d(350,250,400,40030,1);
getch();
closegraph();
}
```

Экранның координаталық жүйесімен жұмыс(Сурет 1):Координатаның бас нүктесі экранның жоғарғы - сол жақ бұрышы.

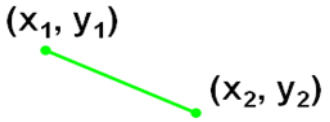


Сурет 1. Экран координаталық жүйесі

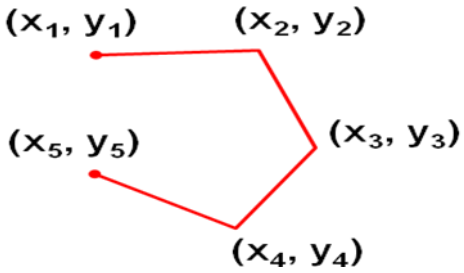
Нүктелер, кесінділер және қисық сызықтар

(x, y)

- **putpixel(x,y)**

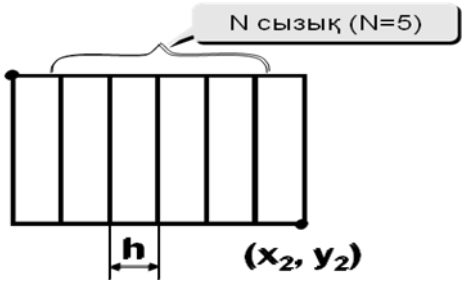



```
Line(x1,y1,x2,y2);
```




```
MoveTo (x1, y1);
LineTo (x2, y2);
LineTo (x3, y3);
LineTo (x4, y4);
LineTo (x5, y5);
```

Штрихтер салу



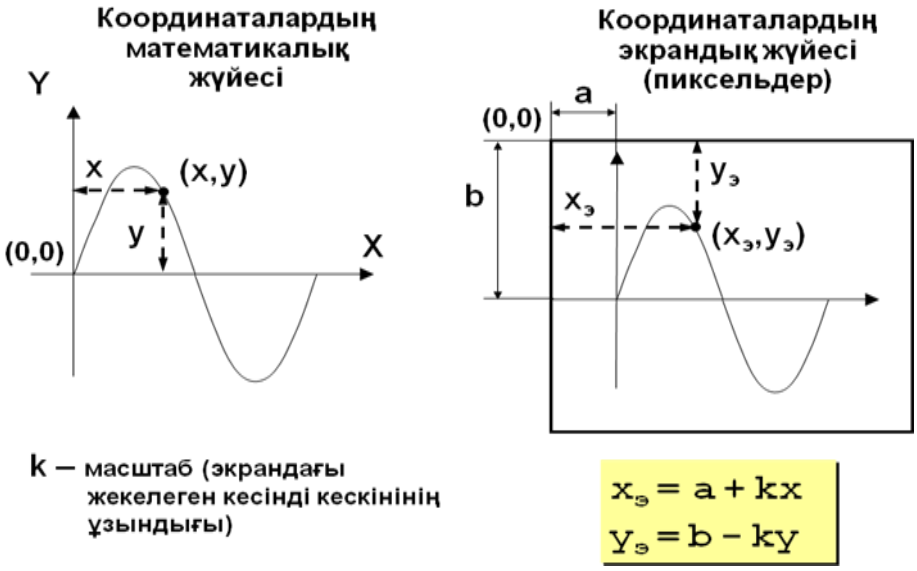

```
Rectangle (x1, y1, x2, y2);
Line( x1+h,   y1, x1+h,   y2);
Line( x1+2*h, y1, x1+2*h, y2);
Line( x1+3*h, y1, x1+3*h, y2);
```

Бұл жағдайда циклдік құрылымдарды қолданған жақсы болады:

```
h := (x2 – x1) / (N + 1);
Rectangle (x1, y1, x2, y2);
x := x1 + h;
for i:=1 to N do begin
  Line( round(x), y1, round(x), y2);
  x := x + h;
end;
```

Математикалық функциялардың графигін сызу

Математикалық функциялардың графигін сызу үшін декарттық координаталар жүйесінен экрандық координаталар жүйесіне өту керек(Сурет 2).



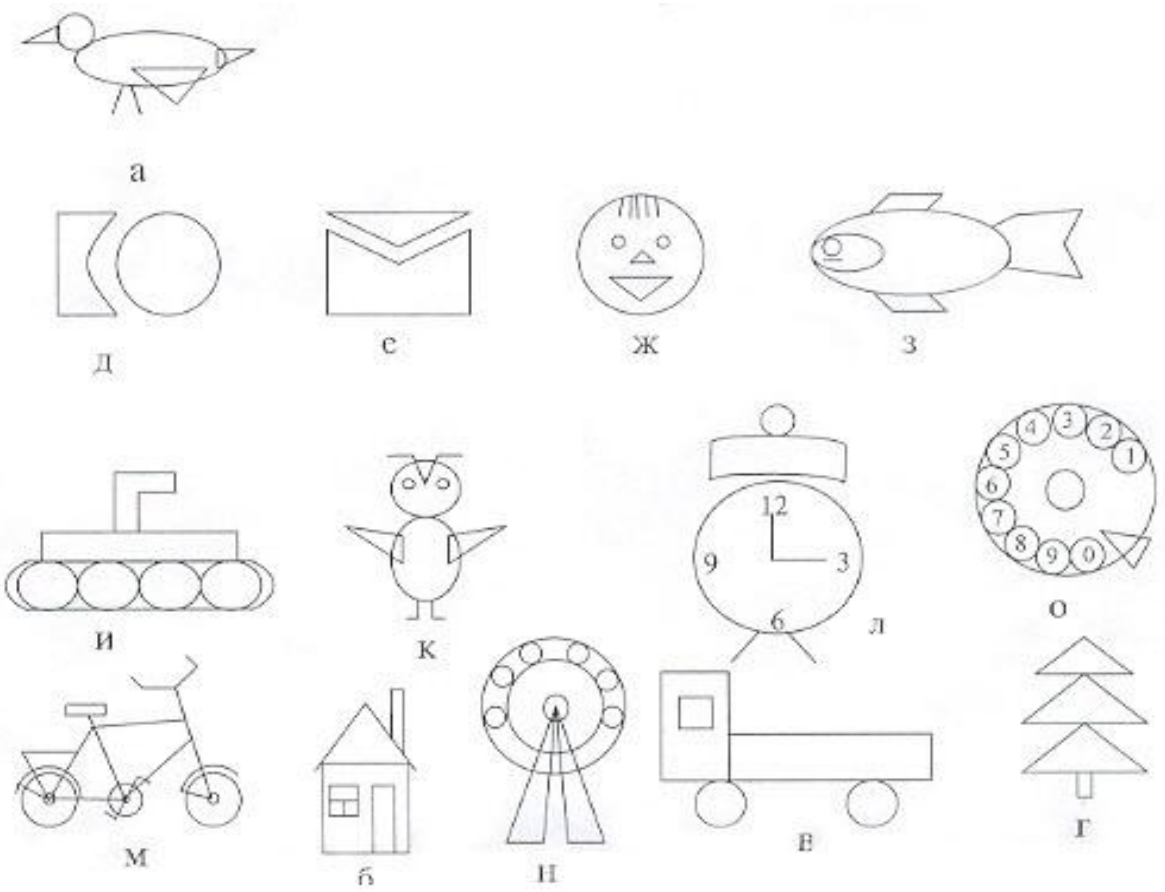
Сурет 2. Координаталарды түрлендіру

Есеп, Экранға sin(x) функциясының графигін сыз.

```
uses  Graph;
var
  Gd, Gm, x1,y1: Integer;x,y:real;
Begin
   Gd := Detect; InitGraph(Gd, Gm, '');
   if GraphResult <> grOk then Halt(1);
      line(10,240,630,240);
       line(320,10,320,470);
   x:=-6.28;
   while x<6.28 do
   begin
      y:=sin(x);
      x1:=trunc(x*50)+320;
      y1:=trunc(-y*50)+240;
      putpixel(x1,y1,4);
      x:=x+0.01;
   end;
     Readln;
 CloseGraph;
end.
```

Тапсырмалар

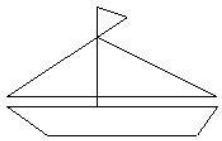
1. Кез келген суретті шығару бағдарламасын құрастыру.



2 Экранда *LineRel* процедурасының көмегімен:

- үйді сызатын бағдарлама жазу керек.

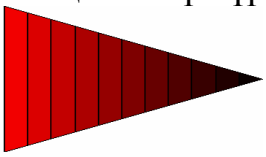
- қайықты сызатын бағдарлама жазу керек.



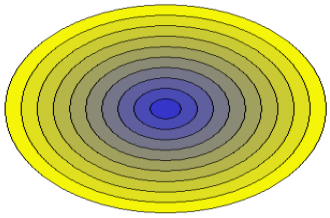
3 Экранға кез келген өлшемдегі және әртүрлі түспен боялған 50 төртбұрышты сызатын бағдарлама жазу керек.

4 Экранға шахмат тақтасының суретін шығаратын бағдарлама жазу керек

5 Пернетақтадан штрих сызықтарының санын енгізе отырып фигура салу және оның ішін әр түрлі түстермен бояу керек.



6 Пернетақтадан шеңберлердің санын енгізу арқылы шеңбердің суретін салу және оның ішіндегі әр шеңберді әр түрлі түске бояу керек.



Бақылау сұрақтары

1 Графикалық режимде экранды басқару функциялар кітапханасы қандай файлда орналасқан?
2 Графикалық объектілерді сызу үшін қандай функцияларды қолданамыз?
3 Графикалық режимде мәтінді шығару үшін қандай функцияларды қолдануға болады?
4 DETECT макросы қандай қызмет атқарады?
5 Экранмен терезер және бейнелермен жұмыс істеу үшін қандай функцияларды пайдаланамыз?
6 Түсті орнату және графикалық объектілерді толтыру үшін қандай функцияларды қолдануға болады?

Қосымша бағдарламалар. Графикалықрежимде массивтерді қолдану.

Есеп-3Р. Декарттық кордината жүйесінде (x,y) нүктелерінің мәндері берілген. Осы мәндерді пайдаланып экранға мысықтың бейнесін шығару керек.

```
uses crt, Graph;
 const  n=36;
 type
      rel = array[1..n] of shortint;
 const
    x: rel=(1,0,2,0,1,0,1,1,-1,0,4,1,3,-1,0,-3,0,-2,-3,
            0,2,-2,0,-1,0,-1,-1,1,0,-1,1,0,-1,0,-1,0);
    y:rel=(0,-2,-2,1,-1,1,1,2,2,1,6,0,2,1,-1,-1,1,2,0,
           -1,0,-2,-1,1,2,1,-1,-1,-4,-1,-3,-1,0,-1,0,-1);
    k=5;
 var
     Driver, Mode,i: Integer;
begin
 Driver := Detect;
   InitGraph(Driver, Mode, ' ');
          if GraphResult <> grOk then Halt(1);
     setbkcolor(3);
       setcolor(11);
         setlinestyle(0,0,3);
          moveto(100,120);
             for i:=1 to n do
               linerel(-x[i]*k,y[i]*k);
           circle(90,120,2);
         Readln;  CloseGraph; end.
```

Есеп- 3Р. (8,8) блокты пайдаланып 0 мен 1 көмегімен объект салып, оны экранға шығар.

```
uses Crt, Graph;
const
 h=10;
 a:array[0..7,0..7] of 0..1=((0,0,0,1,0,0,0,0),
                             (0,0,1,1,1,0,0,0),
                             (0,1,1,1,1,1,0,0),
                             (1,1,1,1,1,1,1,0),
                             (1,1,1,1,1,1,1,0),
                             (0,0,0,1,0,0,0,0),
                             (0,0,0,1,0,0,0,0),
                             (0,0,0,1,1,1,0,0));
 var Gd, Gm,x,y,i,j: Integer;
 begin
  Gd := Detect;
  InitGraph(Gd, Gm, ' ');
  if GraphResult <> grOk then Halt(1);
   setfillstyle(1,1);
   for i:=0 to 7 do
   begin
       y:=i*h;
        for j:=0 to 7 do
          begin
           x:=320+j*h;
           if  a[i,j]=1 then bar(x,y,x+h,y+h);
          end;
   end;
 Readln;
 CloseGraph;
end.
```

Тақырып – 13. Анимация негіздері

Мақсаты: объекттерді қозғалту процедураларын жасауды үйрету.

Алгоритмдік бағдарламалау тілдерінде экранда қозғалысты ұйымдастыруға арналған арнайы процедуралар және функциялар жоқ. Анимация (ағылш. *animation*) – экрандағы кескінге жан бітіру.

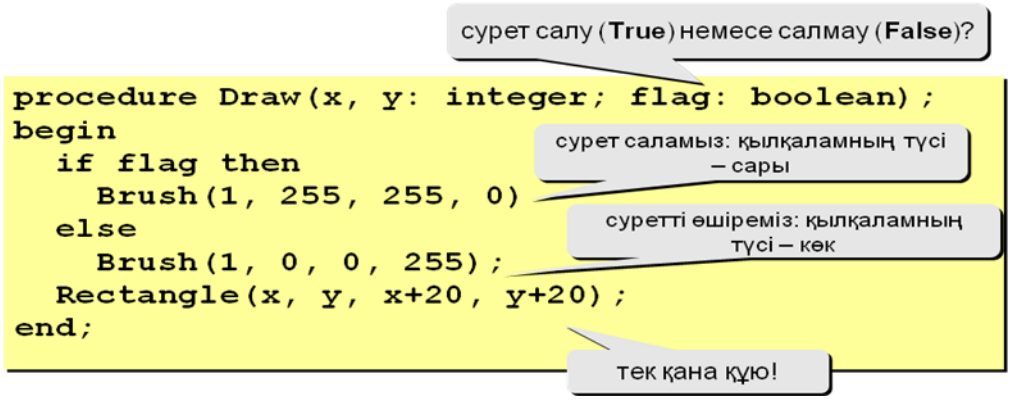
Мәселе: объектті экранда қозғалтуға болады?

Қосымша: объектінің қозғалысы (x,y) координаталарымен белгіленеді.

Анимация қағидасы:

1. (x,y) нүктесіне объектті саламыз
2. (x,y) координаталарын өзгертеміз
3. объектіні жоямыз немесе фон түсімен боямыз
4. осы әрекеттерді бірнеше миллисекундқа кідірту
5. 1-қадамға көшеміз

Процедура (сурет салу және өшіру)



Жыпылықтауды қалай кетіруге болады?

Жауап: ешбір перне басылмағанның өзінде де квадрат әр 20 мс сайын басқа түске боялып тұрады(жыпылықтау!). Бізге қажеттісі: ешқандай оқиға болмаса квадратты басқа түстерге боямау (жыпылықтатпау).

Шешім: квадратты бояп тастап, оқиғаны күту

Алғашқы кезекте экранда қозғалыс жасау үшін қозғалатын объектіні екі рет қолданамыз: бірінші экранның фонының түсімен, екінші басқа түспен ауысып отырады. Осылайша анимация негізі қалыптастырылады. Қозғалыста белгілі сюжеттің негізін қолдану үшін физика – математика пәндерінің заңдылықтарын қолданамыз. Төменде әртүрлі салаға байланысты есептер жинағы ұсынылады.

1 ***Солдан оңға қарай шеңбердің жылжуын көрсететін бағдарлама***

```
Uses Graph, crt;
Var
  X, y, r: integer;{центр, координата, шеңбер радиусы}
  Dx: integer; {шеңбер жылжуының қадамы}
  Dt: integer; {жаңа жерден салуға ұстап қалу}
  grDriver: integer; {драйвер}
  grMode: integer; { графиктік режім }
  grPath: string; { драйвердің орналасуы }
  ErrCode: integer; {графиктік режімнің инициализациялау
нәтижесі}
Begin
grDriver:=detect;
InitGraph(grDriver, grMode, ' ');
   If ErrCode <> grOk then Halt(1);
X:=0; Y:=100; {шеңбердің бастапқы орналасуы}
R:=10; {шеңбердің радиусы}
Dx:=2; {жылжыту қадамы}
Dt:=1000; {ұстап қалу уақыты}
While x<639 do
Begin
 {шеңбер салу}
SetColor(yellow);
```

```
  Circle (x, y, r);
Delay(dt); {ұстап қалу}
SetColor (0); Circle (x, y, r); {шеңбер өшіру}
x:=x + dx; {шеңбер центрінің орналасуын өзгерту}
End;
readln
CloseGraph;
End.
```

2 *Жылжып бара жатқан титаникті салу бағдарламасы*

```
Uses Graph crt;
Const
Dx=5;  Dy=5;
Var
grDriver: integer; grMode: integer;
grPath: string; EveCode,x,y: integer;
Color,BkColor,OldColor: word;
procedure Titanik (x, y: integer; color: word);
const
dx=5; dy=5;
Var
OldColor: word;
Begin
SetColor (color);
Line (x+10*dx, y-5*dy, x+10*dx, y-10*dy);
Moveto (x+17*dx, y-3dy);
Lineto (x+10*dx, y-10*dy);
Lineto (x, y-2*dy); Setcolor (blue); OldColor:=GetColor;
SetColor (color); Moveto (x,y);
LineTo (x, y-2*dy);
LineTo (x+10*dx, y-2*dy);
LineTo (x+11*dx, y-3*dy);
LineTo (x+17*dx, y-3*dy);
LineTo (x+14*dx, y);
LineTo (x,y);
MoveTo (x+3*dx, y-2*dy);
LineTo (x+4*dx, y-3*dy);
LineTo (x+4*dx, y-4*dy);
LineTo (x+13*dx, y-4*dy);
LineTo (x+13*dx, y-3*dy);
Line (x+5*dx, y-3*dy, x+9*dx, y-3*dy);
Rectahgle (x+8*dx, y-4*dy, x+11*dx, y-5*dy);
Rectangle (x+7*dx, y-4*dy, x+8*dx, y-7*dy);
Circle (x+12*dx, y-2*dy, Trunc (dx/2));
End;

Begin
GrDriver:= detect;
InitGraph (grDriver, grMode,' ');
Evecode:= GraphResult;
If EveCode <> grOk then Halt (1);
X:=10;  Y:=200;
```

```
Color:= lightgray;
SetBkColor (Blue);
BkColor:= GetBkColor;
Repeat
X:=X+1; Titanik (x, y, white);
Delay (1000); Titanik (x, y, 9);
Until (x>500);
OutTextXY (10, 10, 'Сапар аяқталады!');
Readln;
CloseGraph;
End.
```

3 *Қозғалып бара жатқан кемелер салу бағдарламасы*

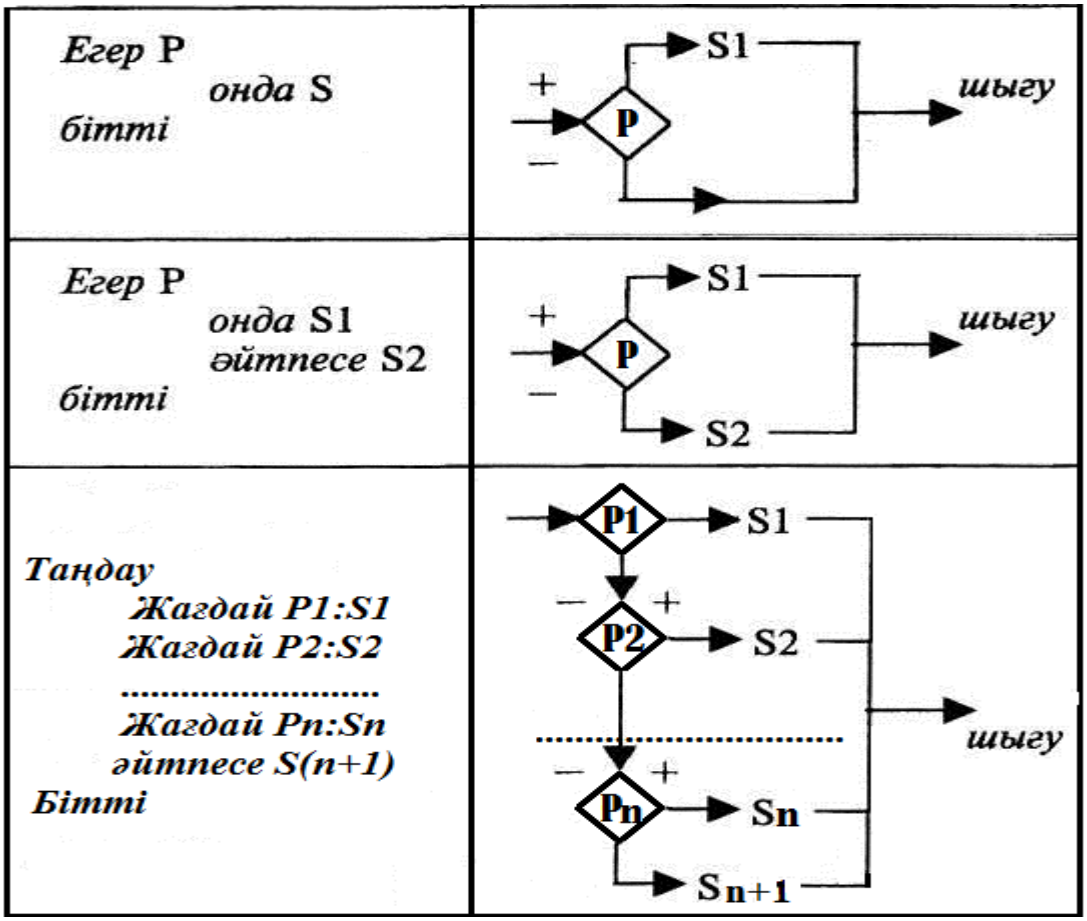
```
Uses Crt, Graph;
Var Gd,Gm:integer;
    X,y,x1,y1:real;
Procedure korablik(x,y,m,color:integer);
Begin
   Moveto(x,y); setcolor(color);
   lineRel(10*m,0*m);lineRel(0*m,-20*m); lineRel(10*m,10*m);
   lineRel(-10*m,10*m);
   lineRel(20*m,0*m); linerel(-10*m,10*m); lineRel(-10*m,0*m);
   LineRel(-10*m,-10*m);
End;
Begin
   Gd:=Detect;
   InitGraph(Gd,Gm,'');
   If GraphResult<>grOk then
     Halt(1);
Line(0,240,640,240);
Line(320,0,320,480);
X:=-2*3.14; readln;
While x<2*3.14 do
   Begin
      Y:=sin(x);
       Korablik(round(50*x+320),round(-50*y+240),1,14);
       korablik(round(50*x+300),round(-
50*sin(x+3.14/2)+200),1,15);
       delay(10000);
       korablik(round(50*x+320),round(-50*y+240),1,0);
       korablik(round(50*x+300),round(-
50*sin(x+3.14/2)+200),1,0);
       x:=x+0.01;
   end;
   Readln;
   CloseGraph;
End.
```

Қолданылған әдебиеттер тізімі

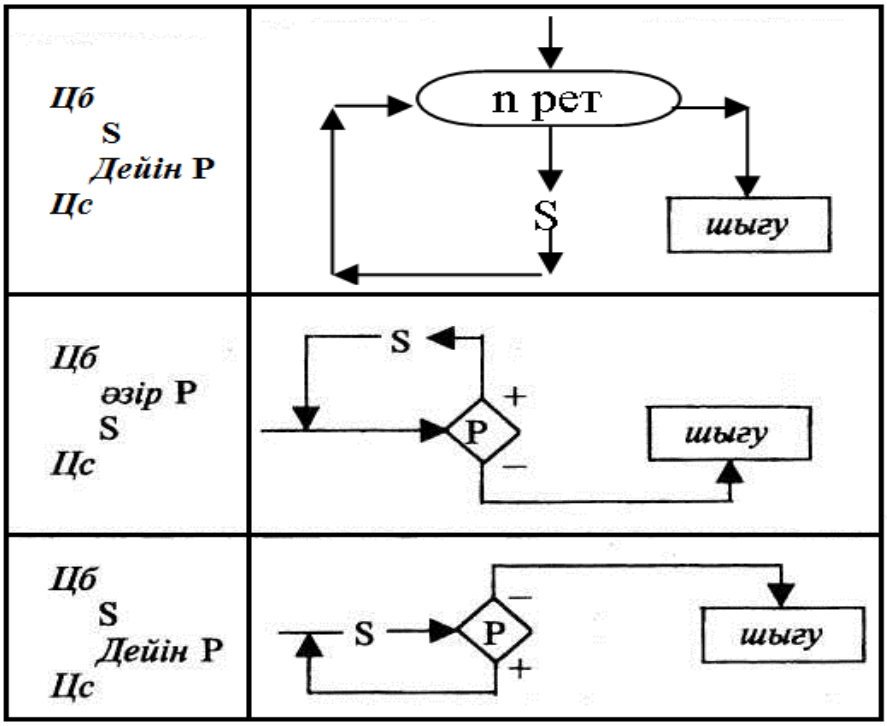
1 Бөрібаев Б. Программалау технологиялары. Оқулық. Алматы: ЖШС РПБК «Дәуір», 2011 -352 бет.
2 Павловская Т.А. С/С++ Программирование на языке высокого уровня/.— СПб.: Питер,2003.—461с:ил.
3 Буч Г. Объектно-ориентированное проектирование с примерами применения. – М.: Издательство Бином. СПб.: Невский диалект, 2013
4 Культин Н. Б. С/С++ в задачах и примерах. Петербург, 2015. — 336 с: ил.
5 Вирт Н. Алгоритмы и структуры данных: Пер. с анг. -М.: Мир, 1989.
6 Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ/ Пер. с англ. Под ред. А. Шеня.-М.: МЦНМО.2002.
7 Сатмаганбетова Ж.З. Бағдарламалау технологиясы. Оқу-әдістемелік құралы. А.Байтұрсынов атындағы ҚМУ, Костанай, 2011
8 В. Фараонов. Система программирования Pascal. – Санк-Петербург, 2003 г.
9 Касаткин А.И. “Профессиональное программирование на языке Pascal. Управление ресурсами: Справочное пособие.” – Минкс: Высш. шк., 2006г.
10 Тумасонис В., Дагене В., Григас Г. Паскаль. Руководства для программиста: Справочник: Пер. с литовск. - М.: Радио и связь, 1992.
11 Сатмаганбетова Ж.З. Бағдарламалау тілі – Паскаль. Оқыту-әдістемелік құралы. А.Байтұрсынов атындағы ҚМУ, Костанай, 2006
12 Юркин А.Г. Задачник по программированию. – СПб.:Питре, 2000

Қосымша А

Алгоритмнің базалық құрылымдары



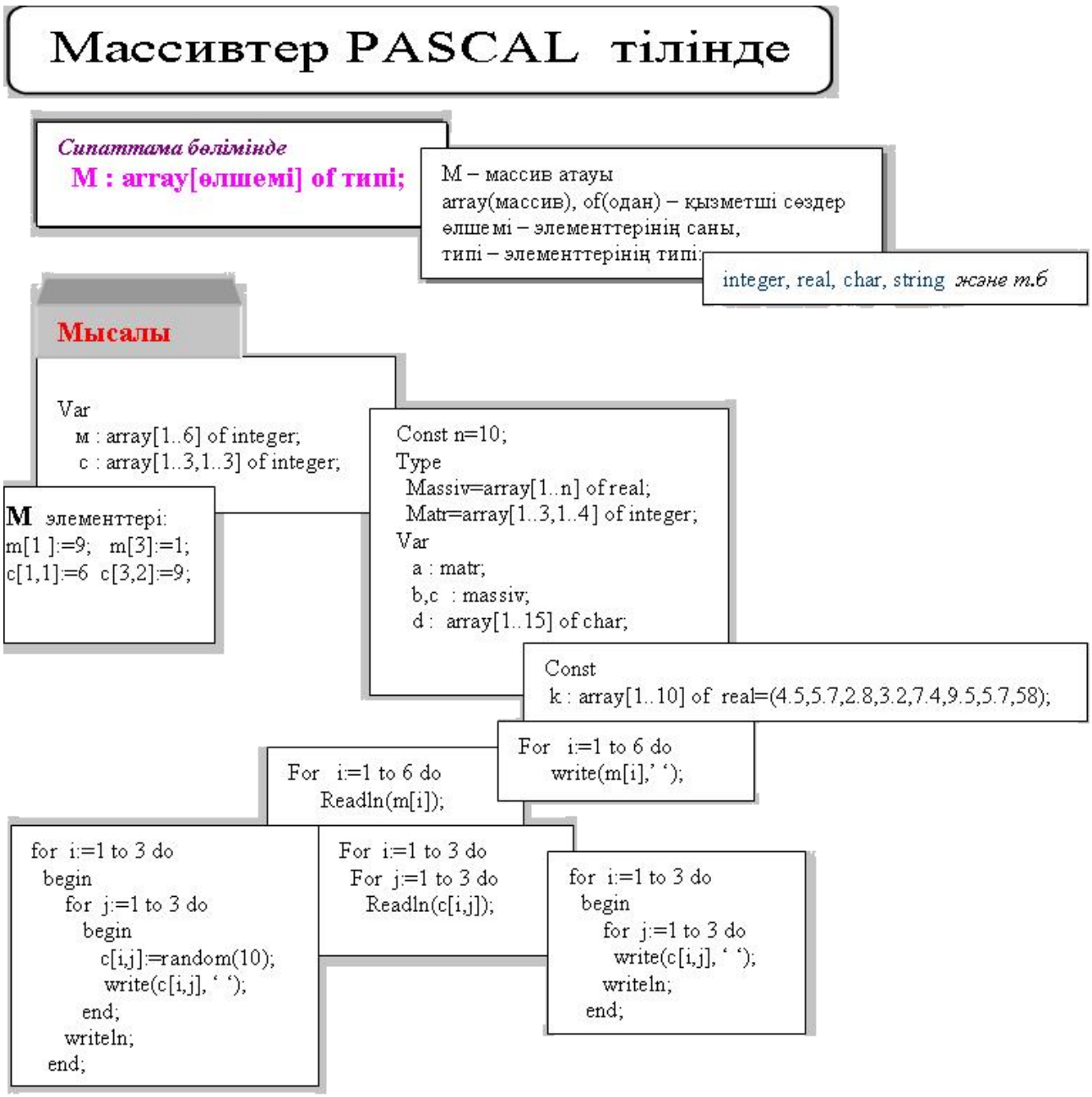
Сурет 3. Тармақталу алгоритмдері



Сурет 4. Циклдік алгоритмдер

Қосымша В

Массивтермен жұмыс



Сурет 15. Массивтермен орындалатын негізгі операциялар

Қостанай - 2017